DOCUMENT RESUME

ED 296 736 IR 052 400

AUTHOR Kahn, Paul
TITLE Isocrates Project. Final Report. IRIS Technical Report 87-2.
INSTITUTION Brown Univ., Providence, RI. Inst. for Research in Information and Scholarship.
PUB DATE 87
NOTE 34p.
PUB TYPE Reports - Descriptive (141)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Classical Literature; *Database Design; *Database Management Systems; Higher Education; *Information Retrieval; Online Searching; *Optical Data Disks; Research Tools; Use Studies
IDENTIFIERS *Brown University RI; System Evaluation

ABSTRACT

A project at Brown University has developed a set of application and system software that provides access to a database of Classical Greek literature which is stored on a CD-ROM. Information on the origins of the project is provided and the technical development of the system is described in detail, including the creation of Classical Greek fonts, indexing the texts, creating the CD-ROM, developing a device driver for the CD-ROM player, adding support for a compact disk file system, and the creation of a menu interface. The delivery of the system to the Classics and Religious Studies department at Brown is then described, along with a summary of the uses of the system in each department. The progress of the system thus far is evaluated, and some suggestions for further development are made. The text is supplemented by 12 figures. (15 bibliographic notes/references) (Author/EW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED296736

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Nicole Yankelovich

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Isocrates Project Final Report

Paul Kahn

Abstract

This report describes the history and accomplishments of a project to develop a set of application and system software which provides access to a data base of Classical Greek literature, which resides on a CD ROM. The origins of the project are described, followed by a detailed discussion of the areas of technical development. These include creation of Classical Greek fonts, indexing the texts, creating the CD ROM, developing a device driver for the CD ROM player, adding support for a Compact Disc file system, and the creation of a menu interface. The delivery of the system to the Classics and Religious Studies departments at Brown is then described, along with a summary of the uses the system has been put to in each department. The conclusion evaluates how well the project has realized its original goals, and goes on to suggest areas for further work.

1. Origins of the project

The Isocrates system is a set of application and system software which provides access to a data base of Classical Greek literature. Most of the data base resides on a single CD ROM (Compact Disc Read–Only Memory). The user can browse the works of a specific author; search for all occurrences of a word, group of words, or a phrase in one or more authors; and copy any text that is displayed into a variety of word processors for later use. The entire system is designed to work on the IBM RT PC under the AIX operating system, though some of the software will run on other computers as well.

Isocrates was, from the very beginning, a synergistic project. The name itself comes from one of the authors in the data base. Isocrates was an Athenian educator from the 4th century B.C. We chose his name for the project because it started with an *I* (at that time all the computers on the network at IRIS had names beginning with *I*) and it was easy to pronounce and remember.

The system was shaped by a number of different forces. In the fall of 1985 our group sought to put together the pieces of a research environment for Classical Greek scholarship, in parallel with the general task of identifying applications software for campus–wide use, part of what became the Faculty Workstation Project. We were aware that the Classics Department had purchased, in the early 1980s, a license for using parts of the data base of Classical Greek literature created and distributed by the Thesaurus Linguae Graecae (TLG) at the University of California at Irvine. This data had been placed on the IBM mainframe in the form of OS data sets. Some custom programming had been done several years before to create a rudimentary linear search application.

We were also aware that Gregory Crane, a member of the Classics department at Harvard University, had developed the Harvard Classics Computer Project (HCCP) software. These programs indexed, searched, and displayed the TLG texts. The programs had been developed under the Unix 4.2 operating system to run on a Digital Equipment Corporation VAX computer. Part of our task would be to make these programs available on the new computer which would be distributed to faculty as part of the Faculty Workstation Project. The computer was the IBM RT PC, though in the fall of 1985 it was still an unannounced product known internally as the Advanced Function Workstation (AFWS). We hoped that making the HCCP programs available on the new computer would give us an immediately useful application. Further we hoped that this would be a motivating factor in moving Humanities computer users, such as the faculty and graduate students in Classics, away from the IBM mainframe to a distributed departmental network of workstations.

We felt that doing this for Classics would give us valuable experience which we would then apply to similar tasks in other disciplines. We assumed that access to the TLG in electronic form would be of general interest to scholars in this field.

The new storage technology of CD ROM offered the kind of storage capacity previously unheard of for "personal" computers. We wanted to be part of the pioneering of this technology, to better see how CD ROM might fit into the plans for computing at Brown.

We also wanted to develop cooperation among researchers at IRIS, M.I.T.'s Advanced Media Lab and Project Athena, and HCCP at Harvard. To this end we contacted researchers at the M.I.T. Advanced Media Lab, and with their permission, made use of an

IR053400

experimental file system for write-once optical disc called Compact Disc File System (CDFS).

The stated goals of the project were limited and technical in nature. In the original project proposal to IBM [7], dated September 20, 1985, we stated that:

> *During academic year 1985-86 IRIS proposes to create an experimental application for searching the text of Ancient Greek authors stored on a CD ROM on the IBM AFWS. This application will be based on the current software developed by Prof. Crane, and will allow the user to search for and retrieve passages of Greek text from the TLG data base, and integrate that text with a document preparation system. The TLG data and index will be placed on the CD ROM.*

The major concern was to get the data base on CD ROM and integrate its use with a document preparation system on the new computer. In a paper written in November, 1985 [9], the project's goals were summarized as follows:

1) *[To] demonstrate the use of an interactive application for accessing a large data base on a CD ROM within the workstation computing environment;*
2) *[To] create a productivity tool by tightly integrating the search and display application with a publication-quality document preparation application;*
3) *[To] test the current theories concerning file systems for the CD ROM by using a large full-text application in a real-world setting.*

This last point was an expression about our own doubts concerning the performance of CD ROM as a storage medium. We knew as little as everyone else about what impact the CD ROM's relatively slow seek time would have on this kind of application. We felt that the best way to find out what that impact would be was to create the application ourselves. For the Classics user, we did not really think that the speed of the application would be a major issue. Where the HCCP software on the VAX could be measured against the use of printed concordance or lexigraphic tools, speed was a comparison of months or weeks against seconds. If the CD ROM made it minutes instead of seconds it did not seem to matter.

There were also less rational forces at work. The entire "scholar's workstation" plan was driven by a vision of what such a computer *should* be able to do. In translating this vision into the context of Classics research there were several images which drove our work. The paper quoted above was expanded into *A Description of Isocrates: All of Greek Literature on a CD ROM*, [10] a paper which was distributed as information about the project beginning in January 1986 and presented for publication at the UCLA conference on Computing in the Classics in June 1986. As stated in this paper, we wanted to make access to the electronic version of the Classical Greek texts transparent to the specialist in this field of knowledge. We saw the speciality of Classical Greek as crossing a wide number of departmental boundaries.

> *The target user community for this kind of application at Brown is a researcher with special knowledge in the field which the data base addresses, but no special knowledge about computer data bases themselves. In this case we are trying to serve teachers and students of Classical Greek texts who would have only occasional use for the data base itself. Such texts are used regularly by scholars in the fields of Classics and Classical Archaeology, and are of potential interest to scholars in History, Philosophy, Linguistics, Religious Studies, and Comparative Literature. Brown's plan for the eventual widespread deployment of workstations makes the assumption that all faculty and research staff will have access to a workstation within their department. Plans also call for the establishment of an infrastructure of local departmental computer coordinators to provide training and support to users within each academic department or unit. Given this context, we wanted to make the application simple to use and easy to support.* [10]

The way in which Isocrates would be used by the various academic departments was seen as part of, and not separate from, the overall plan for deployment of workstation computers at that time. There were also high expectations set for the user interface which was needed to make these electronic texts as easy for the Classical specialist to use as the printed texts in the library.

Another purpose of the project was to become part of and learn more about the working environment of the Classics scholar. We saw the Classics scholar as a specialized writer, and we wanted to make the Isocrates system a useful part of their writing environment.

> *Another anticipated use of the data base is to identify and extract passages of Greek texts to illustrate scholarly writing. [...] We see this ap-*

plication of the TLG texts as the one with the most potential for freeing the scholar's creativity. We want to substitute the Isocrates *environment for a typewriter surrounded by a stack of books marked with slips of paper referring to cross-references scribbled on index cards spread across a desk. The Greek texts will all be stored on the CD ROM. The relevant passages will be located by the* Isocrates *search software, which can store commonly used references in local files. The passages themselves will be extracted by the* Isocrates *software and stored as documents that will appear on the Interleaf WPS desktop display* [see description of Interleaf in section 2.1. below]. *The process of identifying a passage of Greek text, extracting that text into a file on the local disk, and integrating that text into a WPS document has been made as easy as possible. We hope that this ease of use will make it possible for the researcher to access the TLG texts as part of their writing process.*[10]

There is some suggestion of the hypertext model here, and certainly we have had this in mind. But the ambitious goal of truly addressing the information management tasks suggested by this image remained outside the scope of the project. Because of the technical constraints of the software environment, we could not consider exploring the kinds of hypertext research which have been the subject of other activities at IRIS [15].

The hardware and software environment chosen for the Isocrates Project was the IBM RT PC running the Advanced Interactive Executive (AIX) operating system. We considered using the Unix 4.2 version of the operating system on the RT PC; however, AIX, a version of Unix System V, was the only version of Unix commercially supported by IBM. It was clear from the beginning that availability of third-party application software would be limited to AIX alone. AIX was chosen because the availability of commercial software was considered crucial for the acceptance of these workstations in most academic departments. Since the result of the Isocrates Project was to be a specific instance of the kind of workstation computing which the Faculty Workstation Project hoped to generally deploy to the departments at Brown, we selected AIX rather than Unix 4.2 as our development base. Given the context of the project, we needed to accomplish our goals within these constraints.

2. Technical Development

We saw that there were a number of technical problems we would have to overcome to create the system we wanted.

- In order to display, print, and manipulate the TLG texts we had to create Classical Greek fonts for both a document preparation system on the RT PC and for the AIX operating system.
- We had to create indices for all of the TLG texts.
- We had to adapt a file system for use with CD ROM on the RT PC under AIX, and develop a procedure for pre-mastering the data for the CD ROM.
- We had to select CD ROM hardware and develop device driver software for the RT PC under AIX.
- We had to port the HCCP software to the RT PC environment and adapt it for accessing the TLG texts on a hybrid CD ROM/ hard disk system.
- We had to develop a menu interface for the HCCP software that would be simple enough for our target user population.

The following discussion illustrates how we dealt with each of these technical problems.

2.1. Interleaf font development

The document preparation system selected for deployment on the RT PC was Interleaf Workstation Publishing Software (WPS). The Interleaf software is a very powerful document preparation system, designed for use in the technical documentation market. Its strengths include the integration of text and graphics within a document, a sophisticated user-interface derived from the Xerox Star desktop metaphor, and portability across a number of hardware and software environments. Interleaf is an editor/formatter which allows the user to see each page on the screen in a form which approximates how it will appear when the document is printed. The software does this by providing both screen and printer fonts for each available character. To integrate the TLG Greek texts into Interleaf we had to add support for Classical Greek to the Interleaf font library. Interleaf agreed to provide

us with the software tools to do this ourselves. To do font development we needed a stand-alone program called Interleaf Font Editor (IFE) and several utilities, all of which were not part of the Interleaf product. This software ran only on the Sun Microsystems workstation. This software was licensed to us late in September 1985.

2.1.1. Development of Classical Greek fonts on the Sun

Using IFE on the Sun Microsystems 2/120 workstation, we created a set of 10 and 12 point Classical Greek screen and printer fonts. Interleaf provided a copy of the 10 and 12 point Math Greek font that is part of their standard font library. The Interleaf software uses bitmap fonts which are downloaded by Interleaf to the laser printer. This is done to maintain independence from the workstation/printer hardware platform. The term *font* here refers not to a typeface family but to a particular point size and style of that family. The Math Greek family was limited to two fonts: one set of 10 point bitmaps and one set of 12 point bitmaps. Each of these fonts consisted of two corresponding character sets: one set of 72 dot per inch (dpi) bitmaps for screen display and another set of 300 dpi bitmaps for the Imagen 8/300 laser printer used on the Sun network.

Each character in a font was assigned to a position on a key on the keyboard. Characters could be accessed by striking the key in the unshifted, shifted or escape-unshifted positions on the Sun keyboard. This relationship between the key position and a position in the font matrix was fixed in the software. Interleaf provides no "dead key" facility. With a dead key one can assign a diacritic mark used in combination with many vowels to a single key. To produce a vowel/diacritic combination the user strikes this key followed by the key for the vowel. A table lookup then locates the bitmap in the font matrix which corresponds to the combination of this diacritic and vowel. This is the method used by many foreign language word processors on both the IBM PC and the Macintosh, and is a feature actually built into the Macintosh keyboard. Since Interleaf lacked this facility each vowel/diacritic combination had to be assigned to an arbitrary keyboard position. This proved to be a significant drawback for our end-users.

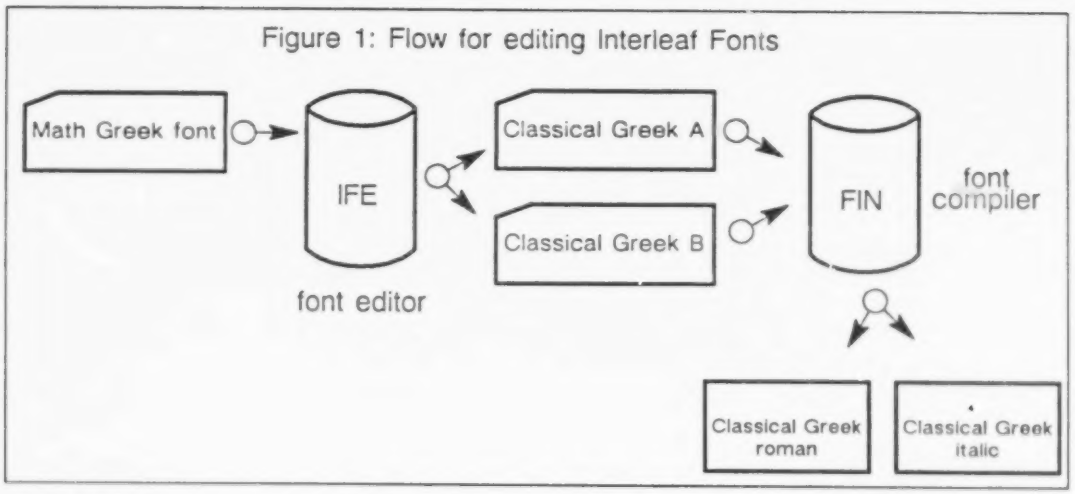
The version of the Interleaf software with which we worked (WPS version 2.0) limited each font to no more than 127 characters. To define what a Classical Greek character set should contain, we used the Greek fonts created for the Macintosh by George Walsh of the University of Chicago as a model. A Macintosh Greek font had been developed at IRIS as part of the Signifier project in 1984, but Walsh's font, sold by SMK under the name *GreekKeys*, better defined the needs of the Classicists. It was already being used by many Classicists who had Macintoshes. The Classical Greek alphabet is actually smaller than the Roman alphabet, but it contains seven vowels, each of which can take up to eleven combinations of diacritic marks. There are also several additional punctuation marks and inscriptional characters used in marking Classical texts. All together the character set contained about 230 individual characters.

Because of Interleaf's 127 character limit, each Classical Greek character set had to be split into two fonts. During development we designated these as the *a* and *b* fonts. Once the fonts were edited using IFE, they were then converted from the editable format to a compiled format readable by the main program using the *fin* utility. They then were written to tape and sent to Interleaf in Cambridge. To be used by the Interleaf software these new fonts had to be recognized by the *fontable* file which provides the font list to the main program. This file is in a compiled format and had to be modified by software engineers at Interleaf. When modifying the *fontable* it was decided that the *a* font would be placed in the Greek roman and the *b* font in the Greek italic slot. The first version of the fonts and *fontable* files was sent back to Brown on tape and installed on the Sun workstation. From that point on we had all the tools we needed to make revisions to individual font files using IFE and install new versions of the fonts in the appropriate place in the font hierarchy.

2.1.2. Porting the fonts from the Sun to the RT PC

The version of TPS on the Sun was structurally similar to the version of WPS released for the RT PC in January 1986. A serial link was set up between an RT PC and the Sun workstation and *uucp* was used to transfer files between the machines. Since the screen resolution of the IBM 6153 monitor is the same as the Sun monitor, the screen fonts and *fontable* files could be transferred directly. The same was not true of the printer fonts, however.

The printer selected by IBM for the RT PC system is the IBM 3812 Pageprinter. This device used an LED print engine with a resolution of 240 dots per inch. The version of Interleaf WPS released for the RT PC would only print on the IBM 3812. This meant that the 300 dpi fonts that had been created for the Imagen 8/300 could not be used on the RT PC. Interleaf supplied us with editable copies of their 240 dpi Math Greek font, and we recreated the 10 and 12



point Classical Greek *a* and *b* fonts at 240 dpi using IFE on the Sun. These fonts were then compiled with *fin* on the Sun and moved to the RT PC. The file-naming convention in the printer font directory on the RT PC version was also incompatible with the *fontable* file from the Sun. This was solved by renaming all the printer font files to begin with "l3" rather than "l". By late January 1986, within a month of the release of the RT PC system by IBM, a usable Classical Greek font had been installed in Interleaf on the Sun capable of printing on the Imagen 8/300 and on the RT PC capable of printing on the 3812.

2.2. AIX font development

Another Classical Greek font had to be developed for use by the application software that displays the TLG texts on the RT PC. To do this, we had to learn about how AIX handles its own system fonts, and find a way to add the necessary characters. We approached this as a general problem, with the expectation that anything we learned would prove useful once RT PCs were deployed to various departments on campus. In retrospect, the creation of a Classical Greek font for AIX has proven to be a unique problem, and no other application for special-purpose AIX fonts has arisen at Brown.

We poured over the pre-release *AIX Operating System Technical Reference* documentation provided to us by IBM concerning font support and the high function terminal (hft) device under AIX. The font support turned out to be very much a part of the Virtual Resource Manager (VRM), and so quite dissimilar to the way fonts are handled in Unix 4.2 systems.

A set of font files and font source files were provided with AIX in the */etc/vtm* directory. The screen manager on the RT console would allow only one font to be used at a time. The documentation explained how a font could consist of up to 1024 characters in 256 character sets. Character sets within a font could be switched by an escape sequence. The source for the font files consisted of raster mosaics in hexadecimal codes and character lookup tables. No font editor was provided.

We developed a plan where we would use IFE on the Sun to create the raster mosaics needed for the AIX Classical Greek font on the RT PC. Using IFE we created a new font that differed from the ones created for Interleaf in size and proportion. This font consisted of 256 characters, all the characters supported by the HCCP software. Since AIX could only deal with monospace characters, each character was fit into a 9 by 20 pixel matrix. From a design standpoint this was actually more difficult than creating the Interleaf screen fonts, since the original Math Greek character bitmaps were all proportionally-spaced character designs. Reshaping each design to a width of no more than 9 pixels was a challenge.

We wrote a program to be run on the RT PC called *fonts* which converted the IFE raster matrix into two binary files: a set of raster mosaics and a character lookup table. These files were then converted to assembler form by a program, *bin2hex*, which converted this binary format into hexadecimal.

Once we had created this raw material we had to manually edit a copy of one of the source font files. The raster mosaic and character lookup table were inserted into the appropriate place in this copy to replace the character set in positions 257 through 512. This file was compiled using the *vcc* and *vrmfmt* programs to produce a file that was ready to be used by the RT PC's control program, the VRM. This com-

piled font file, which we called *grk1.9x20*, was copied into the */etc/vtm* directory. To install a font in AIX it must be defined in the */etc/master* file. Examples of the two modifications to this system control file are shown in Figure 2.

While much of this procedure for compiling and installing a font was described in the RT documentation, there was no program for selecting an alternate font. To select a modified font as the active one in the current virtual terminal, a set of three small programs were developed: *chngfont*, *chngcharset*, and *esc*. These are run in sequence, changing the active font, setting the offset for changing the character set in the font, and setting the escape sequence for shifting in and out. In practice, the sequence of programs with the appropriate arguments is executed from a script which can be embedded in the user's *.cshrc* profile.

2.3. Revising the fonts

This set of one AIX screen font and four Interleaf screen and printer fonts were sufficient for getting starting, but they lacked any clear sense of typographic design. We had taken a set of bitmaps which were intended to be used in mathematical equations and modified them for the purpose of reproducing large amounts of Greek text. There were inconsistencies and imbalances in the letter designs in this context, which were somewhat aggravated by the addition of diacritical marks. The AIX screen font was even more problematic, since in this case the bitmaps had been stretched or squeezed to create monospace characters.

In the spring of 1986, Michael Kroeger, a graduate student in Typography at the Rhode Island School of Design, agreed to revise the font designs. He did this work in August 1986. Figure 3 shows the design for the alpha in all five Kroeger fonts.

In the process of doing major revisions to the font designs, we learned more about how Interleaf manages its screen and printer fonts. The Interleaf software reads the character width values for the printer fonts while displaying the screen fonts. The difference between these two widths is then redistributed among the wordspaces on each line. Given the fundamental difference between the resolution of the screen and printer fonts, there is always some adjustment that must be made. In a paragraph this adjustment is spread over a number of wordspaces, and so is rarely visible to the user. In a single line of text, however, this adjustment can cause the few wordspaces in the line to appear falsely large or small. If the printer and screen font widths are not adjusted properly these wordspaces can even take on a negative value, causing characters to overlap on the screen. Since nearly all the TLG texts are extracted in a line-for-line manner, this made adjusting the screen and printer fonts particularly challenging.

Kroeger used the same software tools, IFE and the IRIS-developed programs for loading fonts under AIX. He revised all the fonts to make them uniformly san-serif in design. Kroeger was asked to revised only the 240-dpi printer fonts for the IBM 3812, since we did not expect any Isocrates users to have access to a 300 dpi printer during the lifetime of the project.

2.4. Working with the TLG texts

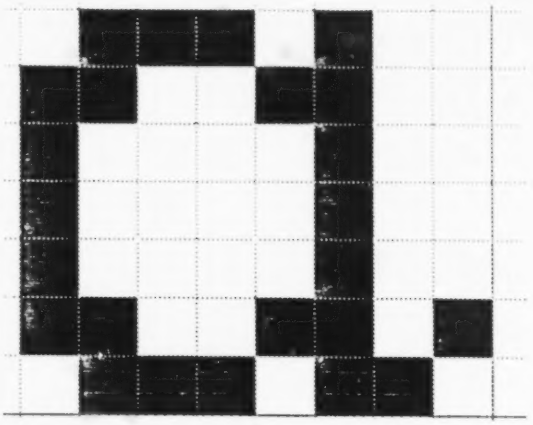
IRIS entered into an agreement with the TLG for producing a CD ROM containing the TLG texts in the fall of 1985. TLG had already agreed to produce a CD ROM containing their texts, and this was in production at the Sony mastering plant in Tokyo. This first CD ROM, later known as the "A" disc, was being made at the request of David Packard, funded by a grant from the David and Lucile Packard Foundation. The A disc had been created using a file format de-

Figure 2: Modified Stanzas in /etc/master (modifications are underlined)

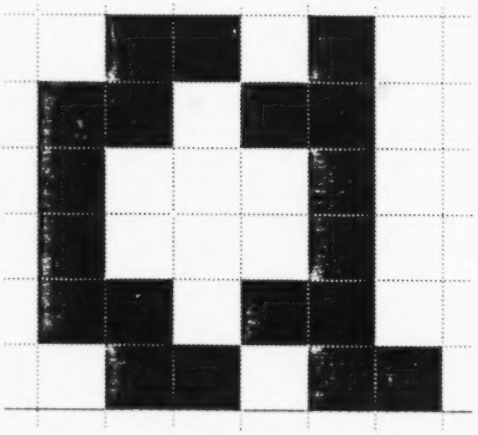
```
uap8f:
            major = 5
            prefix = hft
            mpx = true
            routines = open,close,read,write,ioctl,init
            vdriver = vap8fi920,vap8fb920,vap8fn814,vap8fm48,vap8ft1840,vap8gk920
            config = vrcfont

vap8gk920:
            iocn = 4005
            code = /etc/vtm/grk1.9x20
            ctype = font
```

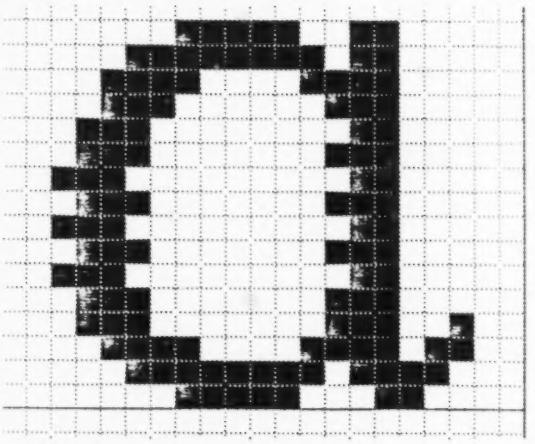
Figure 3: Example bitmaps for Isocrates Greek fonts



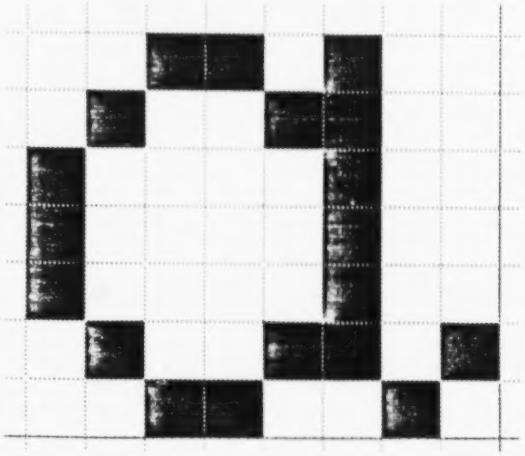
AIX monospace screen font

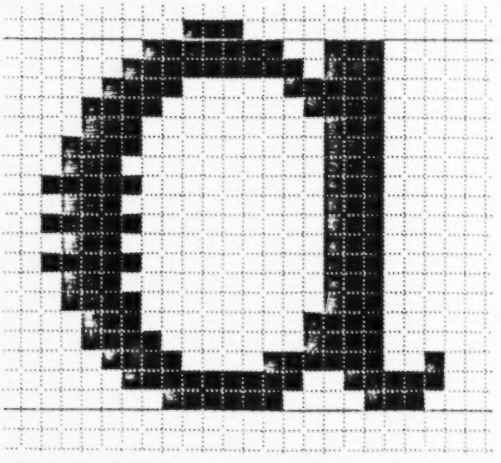


Interleaf 10 and 12 point screen font



Interleaf 10 and 12 point printer font





signed by David Packard, for use with a microcomputer he was designing, known as the Ibycus Scholarly Computer. It contained all the TLG texts, along with a smaller collection of non-Greek texts from the Classical period.

TLG requested that the arrangement between TLG and IRIS duplicate their arrangement with Packard as closely as possible. For the purposes of the project, IRIS wanted to have the CD ROMs produced using the CDFS format developed at M.I.T. We wanted to have control of the pre-mastering process and we also wanted to be able to select the mastering vendor and gain experience in the entire production process. It was agreed that the actual production and ownership of any CD ROM containing the TLG text was entirely the business of the TLG. Rather than have IRIS produce some CD ROMs with the TLG's permission, the TLG would produce the discs and then give IRIS permission to use some number of them. Manufacturing costs for the discs would also be paid for by the TLG from the David and Lucile Packard Foundation grant.

2.4.1. The Nature of the TLG texts

We received a set of seven magnetic tapes from TLG in November 1985. These tapes contained the text of 178 authors, the extent of the TLG corpus that had been verified and released as of June 1985. Throughout the first year of the project we had a rather imperfect idea of what this corpus represented in terms of "all of Greek literature."

In retrospect it is clear that the Thesaurus Linguae Graecae has two parallel missions. The first mission is to define a canon of all the authors who wrote in Greek from Homer up through Late Antiquity (approximately 8th century B.C. to 5th century A.D.). The term *author* in this case means either a personal or corporate author, to use library science terminology (i.e. an identifiable personal author such as Sophocles and an anonymous corporate author such as the compilers of the New Testament are each treated as an *author*). The definition of each work includes the selection of a definitive edition for each work of each author when these works have survived. In fact, most of the authors in the canon exist only as a name in a surviving work by someone else. Another large group of authors survive only in the form of quotation or paraphrase in another author's work.

In June 1985 the TLG canon consisted of 2,884 authors to which were attributed 8,203 separate works. This canon existed as a data base on an Ibycus computer at the TLG facility in Irvine. The first printed edition of the canon had been published in 1977. In January 1986, a revised second edition was published in book form, *Thesaurus Linguae Graecae Canon of Greek Authors and Works* [2]. The above figures for authors and works in the canon come from this edition.

The TLG's second mission has been to create machine-readable versions of the complete works of all existing authors for deposit in what they refer to as the TLG "data bank." This is done by having the text of the approved edition of each author's work keyboarded, using a coding scheme for Classical Greek known as *beta code* developed by David Packard. The keyboarding is done twice by two separate contractors in Asia, and the results are compared using proofreading software on the Ibycus computers at the TLG. All the works of a single author are combined into a single author file. Exceptions are made for particularly large authors (Galen, Simplicius, Origen, etc.); in these cases the author's works are concatenated into two or more author files. Once these texts are verified, they are made available for distribution on magnetic tape. As of June 1985, the 178 verified author files represented 172 separate authors. Every six months the TLG released a new list of authors that have been added to the data bank and were now ready for distribution. The December 1985 TLG newsletter listed 44 new additions. These ranged in size from Georgius Pisides, consisting of 124 words, to Eusebius Caesariensis, consisting of two author files containing over 10 megabytes of text. The July 1986 TLG newsletter listed 98 new additions, and supplements to another 5 previously released authors.

While over 90% of the authors that have been released since June 1985 are so small as to be in the few-hundred-words category, it is clear from these figures that the collection of 178 author files which we indexed and placed on CD ROM for the Isocrates project is not "all of Greek literature." Effectively all the Classical Greek texts that are studied are to be found in these 178 author files, and so this corpus certainly met the needs of scholars at Brown.

2.4.2. Creating the author indices

The indexing of the TLG texts was done on the Sun 2/120 workstation using software supplied by Gregory Crane. These programs, and the index files they create, are documented in Crane's paper *Indexing TLG Author Files* [4], which is distributed with the HCCP software.

In brief, the programs first read each author file from tape into a Unix file. This file is placed in a directory with a name identifying the author it contains. This author name is then used as a base filename for the text file itself. In the case of the works of Hesiod, the file containing all the text is placed in a directory

hesiod, and given the filename *hesiod.tlg*. A script is then run which calls a sequence of programs. The first of these programs parses the text and writes out an unsorted word list, together with a number representing the byte offset for each occurrence of each word. The next program sorts this list, and further programs sort and manipulate the resulting lists, creating a set of indices for that author. When the procedure is finished, the indices contain a sorted word list of that author's vocabulary, information on where each word occurs, and where each work by that author begins.

The question of data verification was very much on our mind at this point. How could we be sure that the texts had been read in correctly and that the indices were accurate? While we could run random tests on the authors by accessing them using the HCCP software on the Sun network, this only proved that the text was there and the index files were functioning. None of the researchers at IRIS could read Classical Greek, and so we could not even be sure that when we did a search in Clemens of Alexandria we were not in fact looking at a passage from St. Basil. The indexing programs created log files and these were searched for any indications of problems. When this turned up problems in several authors the indexing was repeated. The indexing was done as quickly as possible, the programs run constantly on the Sun workstation for 5 weeks during December and January 1986. A new program was added, *newindind*, which we expected would speed up the access time of the search programs. This program created an index to the index, minimizing the amount of binary searching the software would have to do for a particular word in a large index. We were painfully aware that we had no idea how much slower disk I/O would be on a CD ROM, and we wanted to do everything we could to help the matter.

As we did the work, it was clear the job could have been done better on a computer with more CPU power and more disk space than the Sun 2/120 had to offer. Though it took many weeks, in the end we had generated the largest set of indices for TLG texts every created: 126 megabytes of indices for 212 megabytes of text.

The next step was to create the "big index." This involved the creation of the *dbase* directory, containing a file consisting of a sorted list of all the words used in all the authors in the corpus, as well as an index which computed the location of each word in all the authors. This was generated by a separate set of programs which remain undocumented, supplied to us by Crane in January 1986. These programs were run against all the indices we had created, and in turn, created a set of indices that took up another 48.5 megabytes of disk space. This index was derived entirely from the individual author indices, and so relies entirely on these files for its accuracy. It is a redundant feature of the HCCP software, which allows a researcher to locate all occurrences of any word throughout the corpus, without first having to select individual authors to search.

The composition of the final data for the CD ROM is illustrated in Figure 4. We had nearly 400 megabytes of data, a little over half of which were the text files supplied by the TLG. By early February we had all the data we needed to begin pre-mastering the CD ROM.

2.4.3. Regularizing author names

As described above, the HCCP software deals with the TLG text by assuming that a directory exists for each author which contains the TLG text of that author and the necessary indices for that text. The name of the directory and the base name of the index files must be the same, and this name is used to access that author by all the programs.

Author names had been implemented at Harvard by following the Unix philosophy of terseness. If *cp* is preferable to *copy*, then *aesch* was good enough for Aeschylus and *soph* was all one needed to type for Sophocles. At the same time shorter names were not shortened further, so that one typed *homer* to get Homer. This approach seemed unnecessarily obscure. If someone knew they wanted to search for a word in the works of Aeschylus, and they knew how to spell Aeschylus (which seemed like a safe assumption), they should not also have to know that the local abbreviation for that author was *aesch*. Having added more authors, abbreviated names would have added more ambiguity. Would *aesch* stand for Aeschylus or Aeschines?

The assignment of these author names was the task of the person who set up the implementation on any particular system. As long as the author names were consistent on that system, each author could be called anything we wanted. The only limitation we were aware of at that time was the basic filename limitation in all forms of Unix: a file name is limited to a string of characters not including a wordspace or other characters special to the Unix shell. So we could expand *hom_hymn* to *homer_hymns*, but not to *homeric hymns*.

We wanted to use full names for Isocrates, so we looked to the *TLG Canon* as a source for these names. The TLG had regularized all author names by using the Latin form of the name throughout. Where there were more than one author with the same proper

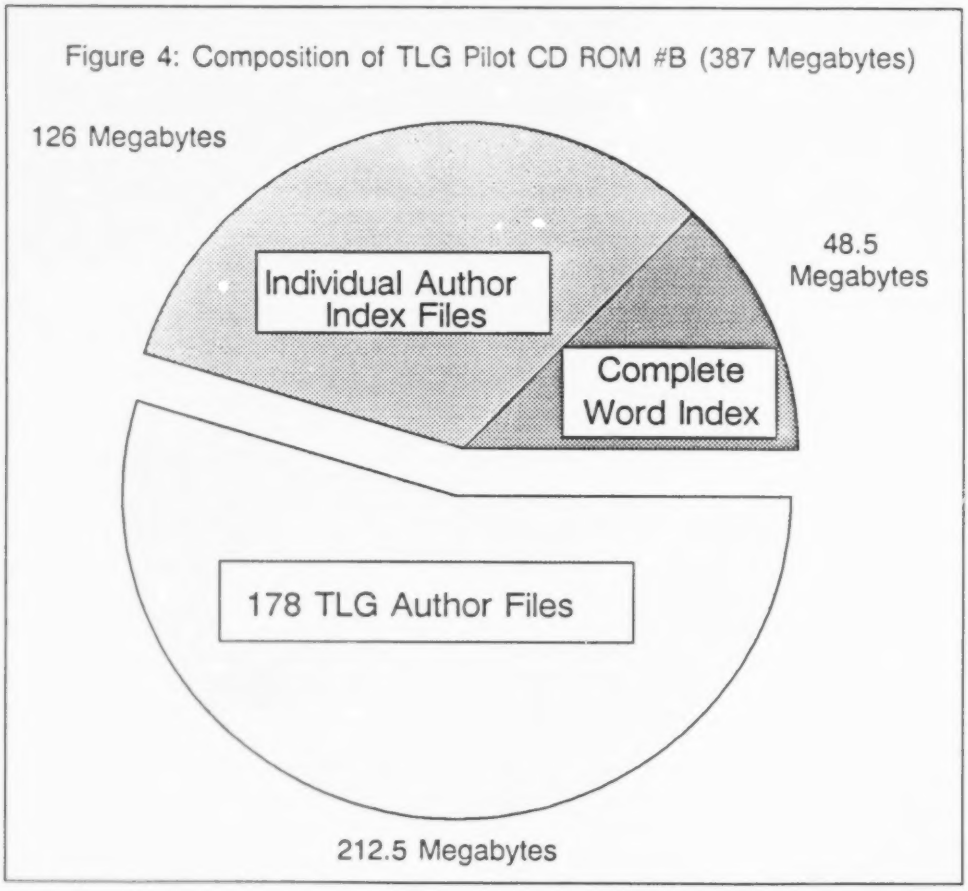


Figure 4: Composition of TLG Pilot CD ROM #B (387 Megabytes)
126 Megabytes
Individual Author Index Files
48.5 Megabytes
Complete Word Index
178 TLG Author Files
212.5 Megabytes

name, something quite common in the *TLG Canon*, the names were disambiguated by a generic and/or geographic epithet.

The system for author names we devised is based primarily on the *Canon*. In those cases where the Latin form of a name simply obscured the common English name for this author, we assigned the more recognizable common name. Two simple examples are using *homer* and *new_testament* rather than the Latin forms *homerus* and *novum_testamentum*. Wherever two words were called for in the name the underscore character was used. The generic or geographic epithet was used only in cases where one seemed necessary to clarify the identity of the author. Plato, the philosopher, was simply called *plato*, and Plato, the comic writer, was given the name *plato_comic*. Where possible the full epithet was included in the form it is given in the *TLG Canon*, such as Clemens of Alexandria as *clemens_alexandrinus*. There were many cases which called for arbitrary decisions. The longest name used was 21 characters, and even with this limit some names simply had to be shortened to something descriptive and unambiguous. Leo Constantinopolitanus Phil. (Leo of Constantinople the Philosopher) simply became *leo_constantinop*. Theophilus Protospatharius, Damascius et Stephanus Atheniensis (the longest "author" name in the list) became *theoph_dam_steph* to distinguish it from the other three "authors" that all began "Theophilus." This list of author names was submitted to several members of the Classics department for review. Elli Mylonas was particularly helpful in adjusting the list to reflect common usage and reasonable compromises.

All this work was being done on the Sun workstation under Unix 4.2, while Isocrates would be run on the RT PC under AIX. In February we became aware of one of the differences between Unix 4.2 and Unix System V: the legal size of a filename. System V recognizes only the first 14 characters of a filename, and this feature was adopted in AIX. AIX does not actually truncate characters after 14, it simply ignores them. This means that an application program looking for the full filename will get it. But if the first 14 characters in the name of two files in the same directory

are not unique, AIX will overwrite the second file with the first. It turned out that a directory with an author names greater than 9 characters could not be copied into AIX without at least one of the names of the index files colliding. There is one collision that occurs at 10 characters and seven that occur at 11 characters. This is illustrated in the sample listing of the files in the *hesiod* author directory in Figure 5 below.

Figure 5: Contents of the *hesiod* directory

```
   26 Mar  1 hesiod.auth
 6432 Mar  1 hesiod.broken
 1077 Mar  1 hesiod.common
 2407 Mar  1 hesiod.comwords
   60 Mar  1 hesiod.file
84589 Mar  1 hesiod.ind
  310 Mar  1 hesiod.odd
35552 Mar  1 hesiod.p2
 3616 Mar  1 hesiod.point
97272 Mar  1 hesiod.prec
16538 Mar  1 hesiod.roman
  105 Mar  1 hesiod.titles
225685 Mar  1 hesiod.tlg
  128 Mar  1 hesiod.workfile
  340 Mar  1 wordlist.bin
   28 Mar  1 wordlist.index
 8182 Mar  1 works.common
```

Since CDFS follows the Unix 4.2 example and allows very long filenames this did not directly effect using the texts on the CD ROM. In this case, it was helpful that CDFS was designed to act as a parallel file system which does not inherit the limitations of the host operating system. But the filename limitation does mean that when copying entire directories of text and index files from the CD ROM to the hard disk, base names would have to be kept down to 9 characters. When it later became clear that we had to have a method for splitting text and index files for authors with long base names between the CD ROM and the hard disk, an inelegant method was devised that allowed us to shorten the base names of these authors on the hard disk.

2.5. Compact Disc File System (CDFS)

2.5.1. Origins of the CDFS

At the time we began Isocrates, the work on CD ROMs done at the M.I.T. Advanced Media Lab was still in its formative stage. Simson Garfinkel, an undergraduate under the direction of Professor Walter Bender, had developed the theory behind CDFS in the spring-summer of 1985. A paper by Garfinkel, *A File System for Write-Once Media*, [3] described the goals of the project. Compact Disc and other optical disc formats under development in 1985 clearly offered enormous possibilities for changing the way in which we used mass storage. CD ROM was available for IBM PC-class machines. The Media Lab had received one of the Sony CDU-1 engineering prototypes. Various forms of non-CD write-once optical disc were being announced at shows, and the Media Lab had indications from a leading vendor that some kind of write-once Compact Disc prototype would be available shortly.

It was this write-once capability for optical disc which was actually the focus of the CDFS project. As described in Garfinkel's paper, CDFS would provide a file system for write-once optical media which would be hardware and operating system independent. This would be done by creating a parallel file system designed for optical disc, a file system that was addressable by various operating systems through a portable software layer. Files written in CDFS format on an optical disc would look like Unix files to Unix, DOS files to MS/PC-DOS, VMS files to VMS, Multics files to Multics, by virtue of a layer of CDFS code, written in C, that stood between the operating system and the CDFS files themselves. This software layer would translate standard system calls to open, read or write files into CDFS equivalents. All the information needed about a file would be stored in CDFS format, and returned to the native operating system upon request. To handle making changes to a file stored in a write-once media CDFS provided a chain of pointers which located the most recent version of each file. Garfinkel actually envisioned write-once Compact Disc using CDFS format functioning as a portable user file system. With a group of machines, each equipped with compatible CD hardware and each running CDFS, a user could store all versions of his files, a complete history of his work, on a single CD and carry the CD from machine to machine. With 550 megabytes of storage on each CD, nearly 500 times the storage capacity of the largest other portable medium available at that time, the possibilities seemed enormous.

But by the fall of 1985, the prototype for a write-once Compact Disc had not materialized, and indications from all the vendors were that such a device was 12 to 18 months away. CD ROM was very much with us, and so work at the Media Lab turned to using CDFS as a way to master CD ROMs. This was precisely our main purpose in adopting the CDFS system for Isocrates.

The original CDFS code was developed on a VAX-11/780 running Unix 4.2 BSD. There was no optical disc drive of any kind available for the VAX under Unix. To overcome this limitation, Garfinkel simulated the existence of a write-once optical disc device in software. This part of the code, known as the Simulator, was used for testing the functionality of the design and later for pre-mastering.

This version of CDFS, the code we first worked with in the fall of 1985, had two serious limitations. Binary information was stored in a way that was specific to the byte order of the VAX hardware. This meant that an implementation for the MC68000-based Sun workstation, or the as yet unannounced RT PC, would require an enormous amount of byte swapping in the code.

Second, to pre-master a CD ROM, one had to write out the data in CDFS format on standard magnetic hard disk using the Simulator. The original code for the Simulator was limited to addressing a single logical device. This meant that a CD ROM mastered in this manner could contain no more data than would fill a single hard disk on a VAX, a little over 400 megabytes. The first CDFS CD ROM was mastered in this manner at the Media Lab in October 1985.

To use CDFS at IRIS, we would have to get the Simulator code running on the Sun workstation. Garfinkel saw the byte order problem as a weakness in the CDFS design. By October he was able to rewrite the code so that it was byte-order independent. We then needed to have sufficient space on the Sun network to first create the text and index data, and then format this for CD ROM using the CDFS Simulator. The largest physical device on the Sun network, the Fujitsu Eagle, only gave us 350 megabytes of usable space. Garfinkel corrected the single-device limitation as well, and the version of the CDFS code we had running on the Sun by November was able to store Simulator files across physical devices using soft links. We arranged for about 1 gigabyte of disk space across several partitions for use during the project.

At the same time the task began of porting the CDFS code to AIX on the RT PC. This proceeded slowly for several reasons. We did not receive a final version 1.0 of AIX until January 1986. The CDFS code was written to run under Unix 4.2, and translating the system calls to AIX, based on Unix System V, was complex and time consuming. We also found that the AIX C compiler was quite unforgiving compared to the 4.2 compiler. The work was done on a part-time basis by a Brown undergraduate employed by IBM ACIS. This effort also became entangled with our early attempts to create an AIX device driver first for the Sony CDU-1 and then for the Philips CM100. These efforts were only partially successful and they delayed completion of the full CDFS port.

2.5.2. Media Lab's first CD ROM

The first CD ROM created with the CDFS format was pre-mastered at the Media Lab by Simson Garfinkel early in the fall of 1985. This CD ROM contained a mix of text, binary, and image files. It's purpose was to be an existence proof that a CD ROM could be created in this manner. It was also intended for testing access time using the CDFS code on an IBM PC/AT, since this was the only computer available with a hardware interface to a CD ROM player at that time.

The manufacturing was done at the 3M plant in Menonomie, Wisconsin. Technical problems came up concerning the format in which the tapes were to be written. CD ROM manufacturers required that the data be delivered on magnetic tape in one of two forms: VMS dump format or a tape with a specific form of ANSI label. The data was on a Unix VAX which did not have any standard software for creating either of these kinds of tapes. Garfinkel found a program, *ansitar*, which had been designed to write tapes in ANSI format using the syntax of the Unix *tar* command. He modified this program to write the tapes.

There was a minor mistake made on this first CD, due to a misunderstanding of the Philips/Sony standard. This standard requires that the first 2 seconds of a CD be left blank. Garfinkel interpreted this to mean that he should supply zeros in the first 2 seconds of data. The mastering program added another 2 seconds of zeros, so the location of all the data on this first CDFS disc was 2 seconds off. Since this put the location of all files off by a consistent factor of 2 seconds, Garfinkel could and did adjusted a version of his code to account for this mistake.

The Media Lab CD ROM was read on an IBM PC/AT equipped with the Sony CDU-1 CD ROM player. No one was impressed by the performance of this player, but it worked. The PC/AT was fitted with an ethernet board and connected to the Unix VAX. Garfinkel got to work on an addition to CDFS which allowed CDFS code on the VAX to use the PC/AT as a file server. CDFS code running under DOS served the CD ROM data to the CDFS application running under Unix on the VAX via the ethernet link.

2.5.3. Working with the CDFS Simulator at IRIS

While this work was going on at the Media Lab we began the pre-mastering effort at IRIS using CDFS on the Sun. We wanted to use the Simulator to create a

sequence of virtual CD files containing all the data needed for the Isocrates application. Once again there was no documentation for this software. Work was done based on Garfinkel's directions.

The first step was to compile a set of executable programs that would create and manipulate CDFS files in the Simulator format. Two files in the *CDFS/Simulate* directory define the parameters for a specific CD ROM. The *CONFIG* file defines the number of virtual CD files that will be created, their sequence, and their location in the file system. Each file was set up to contain 7 minutes of data, the amount that would fit on a single magnetic tape written at 6250 bpi. The *HARDWARE.h* file defines the four parameters used to translate bytes into Compact Disc playing time: number of bytes per block, number of blocks per second, number of seconds per minute, and number of minutes per disc. This file is referenced at the time that CDFS creates the superblock for the virtual CD.

The CDFS *makefile* is designed to allow the system to be made for a number of CD devices. The driver for each device is bound into the code during the compile. The device (in this case the Simulator is considered the device) is selected by passing an argument to the *make* program.

The program to initialize the virtual CD is *MAKECD*. This prompts for a signature line to be written into the superblock. It then creates the first *VDISK.00* file. Space was allocated for the *VDISK.xx* files by creating them in directories where there was sufficient free space, then linking them to a single directory using the Unix *ln -s* command.

The TLG data was copied into CDFS format using the *cdput* command. This takes a Unix file and translates it into the 2K block format used by CDFS. The *cdput* command supports a recursive flag (*cdput -r*) and this was used to copy entire directories using scripts. CDFS provided a set of Unix-like utilities for examining the files in the CDFS format. The location and size of the files could be checked using *cdcd*, *cdpwd*, *cdcat*, and *cdlist*.

Once the *VDISK.xx* files were created, each file was written to tape using the *ansitar* program. This produced a set of seven tapes. 3M was selected as the vendor to do the mastering because they had already successfully mastered a CD ROM using this format.

Despite these precautions, we had technical problems getting the discs through the manufacturing process. The problems were due once again to misunderstanding, in this case concerning the ANSI label format required for the tapes. Three sets of tapes had to be cut at IRIS before we were able to create the tape format required by 3M, costing us about six weeks in delays. The master was finally cut in March. The 50 discs were shipped to the TLG at U.C. Irvine in April. We requested and received 5 copies of the disc, known as the **TLG Pilot CD ROM #B for Experimental Purposes**, later that month.

2.6. Creating the CD ROM device driver

It is important to remember just why we insisted throughout the project that the entire application run on the IBM RT PC. This machine was to provide the hardware and operating system environment within which we were to define the "scholar's workstation." We postulated that this required certain things that were not to be found in the IBM PC/AT or the Apple Macintosh. The IBM RT PC did not meet all of these requirements either, but we had no other alternative as far as hardware platforms went. We selected AIX rather than 4.2 for the operating system, trading a supported operating system and commercial application software with minimal networking and an 8 inch monitor (AIX) for a network file system, 19 inch monitor, a window manager, and no commercial applications software. This decision was made because AIX was deemed more appropriate for deployment to the faculty departments and Isocrates was intended to be used in the departments.

A CD ROM player was a new device and we knew that it would require a device driver. We had significant expertise at IRIS in doing system programming for the RT PC under Unix 4.2. We did not know what AIX would require that was different from Unix 4.2, and we did not know how a CD ROM drive differed from other disk devices. There was nothing that looked to be ground-breaking in creating a device driver for a CD ROM player, certainly not compared to other work being done at IRIS such as adding support for window managers and network file system in the 4.2 kernel. In retrospect, dealing with the hardware and software interface to the CD ROM player from the RT PC turned out to be the most difficult technical part of the project.

2.6.1. Choice of the Philips CM100/CM155

We began by doing some limited testing with the Sony CDU-1 engineering prototype. This device consisted of the CD ROM player, a separate power supply, and an IBM PC interface board. We did not seriously considered using this device for the purposes of the project, but it seemed like the best route to learning what would be involved in getting a CD ROM player working with the RT PC.

With this in mind we did the preliminary work of defining a CDFS device driver for the Sony CDU-1

on the RT PC under AIX. When enough of the CDFS code was ported to AIX to support the *cdlist* command, we were able to use CDFS to read directories off the Media Lab CD ROM using the Sony player. But anything more complex, such as using *cdcat* to copy an entire file from CDFS to AIX, would not work. The hardware interface was too simple to handle high-speed data transfer.

Through contacts with Philips at various conferences and trade shows we decided that the Philips CM100 with the CM155 interface board would be the best production unit to use for the project. The people at Philips were willing to help us, and they were all located in the U.S. We purchased one CM100/CM155 set and began to work with it. We later purchased six more.

2.6.2. RT PC/CM155 hardware problems

The CM155 was designed for the IBM PC, and the documentation which came with it assumed that a programmer was starting from MS/PC-DOS. Philips supplied a DOS device driver, without source code. The standard programmer's documentation supplied little information on the CM155 hardware.

We first tested the unit on a PC/AT using some DOS utility programs supplied by Philips, then with a copy of the Grolier Electronic Encyclopedia CD ROM. Attempts to run the same software on the RT PC's AT coprocessor failed completely. The RT PC did not appear to be able to get a signal to the CM155 board.

All attempts to make the CM155 respond on the RT PC proved futile. Once we had determined that we could not even write to registers on the board under DEBUG mode using Unix 4.2 on the RT PC we sought help from the engineering support people at Philips. The CM100/CM155 worked fine on the PC/AT and did not work at all on the RT PC under Unix or under DOS on the AT coprocessor. One possibility was some fundamental difference between the bus structure on the two machines. Philips indicated that the way in which the machine handled the Address Latch Enable (ALE) signal was crucial to the board's performance and they had already seen some variation in this among PC/AT clones. We poured over the technical reference manuals for the RT PC, in particular as to how they described the ALE signal. IBM did not provide a simple summary of differences between the AT and the RT bus structures. We did not at this time have any contact with engineers at IBM Engineering and Scientific Products (ESP) in Austin who could have answered such questions.

In the spring of 1986 we received help from Michael Johnson, an IBM engineer assigned to Project Athena at M.I.T. It was also in Project Athena's interest to determine what the differences were between the RT and the AT bus, and in particular what modifications needed to be done to make the CM155 board work on the RT. The engineers at Philips were willing to make modifications to the boards we had purchased at no charge, but they did not have access to an RT PC and so could not determine what the source of the problem was.

Johnson isolated two general problems, both of which effected how the CM155 performed on the RT PC. He described the problems this way:

> *The CD ROM card like the Metrabyte board* [a data acquisition card also being tested at M.I.T.] *only decodes 10 bits of I/O address, instead of the full 16 bits. Since DMA registers are at 88xx and 8Cxx, this will cause havoc.*
>
> *On the RT one of the control lines ALE (address latch enable) is always high. This causes a problem for the CD ROM card because there exists a programmable latch on the card itself that is strobed when ALE goes low. This programmable latch sets the mode of the card, and initializes it. Since the card's mode is never set and initialized, communication with the card is impossible in an RT.* [12]

This defined the two hardware incompatibilities to be overcome. The fact that only 10 bits of I/0 address were used on these boards was due to the fact that the IBM designers of the PC/XT/AT only used 10 bits of I/0 address in their implementation. The actual IBM specification for the AT bus called for 16 bits of I/O address, and the engineers at IBM ESP chose to follow this specification rather than the actual PC/XT/AT implementation when designing the RT bus. This meant that data transfer using DMA could work with the AT coprocessor, but would be impossible on the RT's RISC processor without some kind of "transformer" hardware that we were not prepared to design. The CM155 was designed to support data transfer using either DMA or programmed I/O. The Philips device driver used the former, but we had to use the latter for data transfer on the RT PC.

Once the nature of the problem with the ALE signal was clear to Philips engineers, they modified the CM155 board so as to completely circumvent this signal for initialization and mode selection. Using these boards specially modified for the IBM RT PC, we began to develop the device driver software.

2.6.3. AIX user-level and Unix 4.2 kernel-level drivers

We first developed a user-level driver to run under AIX. To test it, we created a program, *cdtest*, which could read a number of blocks of data on the CD given a minute/second/frame location. It was found that the new problem was timing. The DMA interface provided for error checking in hardware, but this was not true for programmed I/O. To read a block of data from the CD using programmed I/O the program had to first see the header block, then read the data, then read the sync block and check for data accuracy. At the heart of the CD ROM player there was a fixed timing requirement. The driver had to be there to read the data every *n* microseconds. If the driver was interrupted by anything, even the hard clock, the timing was off, the sync block was missed and the data was lost. There was a register on the CM155 that could be checked to see if the data had been transferred, but it too was timing-sensitive. The character buffer on the board was too small to be of any help.

At this time we had made the first modifications to the HCCP programs to make them work with the CDFS code. We compiled this user-level driver for the Philips CM100 into the CDFS code. For the first time we were able to search an author off the CD ROM and get results on the RT. Trying to transfer any large amount of data resulted in many retries due to timing problems. We did not expect that a user-level driver would be a final solution, and this certainly proved us correct.

It was clear that to overcome the timing problems the software would have to be able to block all interrupts during a read. We went back to Unix 4.2 to regain control of the kernel and the hardware, to verify an algorithm in this controlled environment. To solve the timing problem we developed an algorithm that used the RT PC's machine cycles for a timing loop. After some trial and error we came up with the right number of machine cycles. The *cdtest* program was rewritten to work under 4.2 and a kernel was built with the driver. This program was able to read large blocks of data off the CD ROM. Work then shifted to AIX.

2.6.4. The VRM device driver

We had already dealt with the issues of porting software from Unix 4.2 to AIX when moving the HCCP and CDFS programs to the RT PC. The differences between the operating systems were primarily those of details and names of system calls. The differences between how these two versions of Unix handle hardware devices proved to be far more significant.

In developing their RT PC product, IBM ESP had followed the model of the IBM 370 software architecture and placed a software layer between the hardware and the operating system. Under Unix 4.2 the operating system has direct control of all hardware devices. Under AIX all control of the hardware is done through requests to the Virtual Resource Manager (VRM). The VRM did not resemble any version of Unix, and in fact was much closer in design to VM/370. IBM did supply documentation on how to add devices to the VRM, but we soon found that this documentation, and the procedures they outlined, were untested. No third-party developer had yet created an AIX device driver. The examples supplied in the documentation were all written in C, but engineers at IBM ESP actually used the PL/8 language for all their internal development work.

We received a revision of AIX, version 1.1, in September 1986, and began to study the new documentation to determine how such a driver could be implemented under AIX. Under AIX all device drivers are divided between the VRM and the AIX kernel. Once we had the basic files laid out we implemented the algorithm for the read loop under VRM/AIX. First a driver was designed for AIX, then for VRM. In this case the AIX driver did very little, since most of the work was hardware manipulation. All calculations were being done by VRM. The AIX driver was similar to the 4.2 driver in design. The modules were the same with all the 4.2 code converted into calls to VRM routines. Then the VRM driver was written to provide these routines.

In VRM a driver is not called by an AIX driver directly. AIX calls VRM, then VRM calls the driver. This is done by SVC (supervisory) calls from AIX to VRM. VRM acts as an interface queuing system for the hardware. Once the driver receives the call, VRM passes it necessary information depending on how the AIX driver requests the call. VRM translates the call into its own structure and passes the call to its hardware driver.

The driver for the CM100/CM155 under AIX consisted of the following files and programs:

/etc/predefined: This is a copy of the standard */etc/predefined* file which contains a description of all available devices on the system, with a description of the CM155 board added. This is used by the *devices* command to create */etc/system*.

cdadev: This is a program which is run twice to add two stanzas to */etc/master*. These stan-

zas supply information on AIX and VRM drivers for CM155 to the VRM.

/etc/configcd: This program is used by *devices* to configure the CM155 into the VRM

/etc/vrmdd/vrmcd: This is the actual VRM driver.

/etc/ddi/cdrom: This is the file which contains the hardware parameter for the CM155 card which *devices* uses to put information in the VRM.

/etc/ddi/cdrom.kaf: The information in this file tells the VRM and AIX what to do with the device when it is added, deleted, or the system is IPL'ed. It also can contain information used by the *devices* command to change certain hardware parameters, but this feature is not used in the case of the CM155.

u155cm.o: This is the AIX driver which is linked into the kernel.

These files were put in place and the programs run using a script, *cdinstal*. The script adds the CM155 to the system using *devices*, rebuilds the AIX kernel and reboots the system. Once this was done the *vrmconfig* program is run to dynamically configure the device into the system. This program calls the VRM and tells it what it wants to configure. It supplies information about the VRM driver, identifies what is device-dependent, and puts all this into the system data structures. The configuration process *glues* the VRM driver code to the VRM nucleus. This puts the code into VRM address space. It is *glued* to bind it dynamically rather than statically. This means you don't recompile and link the VRM to permanently attach the code to it. This feature of dynamic binding is unique to the VRM as far as we know.

While it did work, the driver was not stable. We found that a number of random activities would cause the RT PC to crash. These appeared to be caused by conflicts between the CD ROM device driver and other code in the VRM address space. From November 1986 until February 1987, we continued to use this unstable version of the driver, experimenting with various changes to the programs. The crashes occurred in the VRM, and the debugging facilities provided in that environment did not give us enough information to locate the cause. The final solution turned out to be adding the device driver only when the CD player is to be used, and then deleting it from the VRM when it is inactive. Once this procedure was implemented the system proved to be completely stable.

2.7. Adapting the HCCP software for Isocrates

When we began in July 1985 the HCCP software consisted of files arranged in the following set of subdirectories:

tlgsend/auto: source files used to create the automaton which parses TLG beta code and translates it for output to the supported devices;

tlgsend/lib: ASCII files used by other programs;

tlgsend/prepsrc: source files used to create the programs for indexing the TLG texts;

tlgsend/scripts: shell scripts used by programs and users;

tlgsend/src: source files for the search and display programs.

The *tlgsend/src* directory contained four executable programs: *tlg_sel*, *tlg_rec*, *tlg_exp*, and *tlg_new*. These programs are documented in the HCCP *User's Guide For Search Programs* [5].

2.7.1. Porting to AIX

Porting the basic code to AIX was done quickly. The code had been designed to be ported, and had already been ported to the AT&T System V Unix that runs on the 3B2 series. Our goal throughout was to maintain a version of the HCCP code that would work on all the Unix platforms, not create a customized version for the RT PC. The most complex problem was figuring out how to get the RT PC monitor to provide VT100-like scrolling areas. The solution was to clear the scroll area and draw down from the top each time you wanted to "scroll" forward or backward. We had to modify the screen-handling routines in *tlg_rec* somewhat to make this work on the RT and other devices uniformly. All the basic HCCP programs functioned under AIX by September 1985.

The next step was to add support for the new devices that would make up the Isocrates system: the Interleaf Greek fonts (ileaf), the RT PC's bitmap display monitor (hft), and the Signifier Greek fonts for the Macintosh (signf), a Classical Greek font developed at IRIS in 1984.

Support for a new device requires adding a table in the *tlgsend/auto* directory which defines the value for each Greek character for that device. Entry points to this table also have to be added at several places in the code. The table is then compiled into the automaton library. The automaton parses the TLG beta code of the text file and returns the name of a legal Classi-

cal Greek character for each character or character string. The translation program then writes the value of that character to output (standard output or a file) by selecting the value from the table for the specified device.

Support for the hft device was handled just like support for MacTerminal, and the Signifier Greek font was added after the model of the GreekKeys Greek font for the Macintosh. There was no direct model to follow for adding support for Interleaf, however. The Interleaf characters were specified in the table in a standard manner, but the Interleaf ASCII loader, that part of the Interleaf system which would read the resulting file and format it on the Interleaf desktop, needed some additional information.

There needed to be an explicit "hard return" code at the end of each line of TLG text in order to maintain the line-for-line appearance of the original. This had previously been handled by placing a new-line character at the end of each line, but that was not enough in this case. Interleaf needed to see a character string, *<HR>*, at the end of each line.

Interleaf also needed a file header containing the name and definition of all fonts and components in a file. This information needed to be appended to the top of the file created by the HCCP software. To appear as a document icon on the Interleaf desktop, this file needs to be located in the *desktop* directory of the user's home, and given a filename limited to no more than 6 characters followed by a *.doc* extension.

We modified the format of the table source files in *auto* to include two new parameters. One of these, *xxx_newline*, allowed us to specify a character or character string for the newline function in each device. The second, *fix_xxx_save*, allowed us to specify the name of a file in *tlgsend/scripts* containing a script that is run when a file is saved for this device.

For Interleaf a script was added, *tlgsend/scripts/ileaf*, which concatenated the file being saved in Interleaf format with a header file stored in *tlgsend/lib/template.ileaf*, placed the resulting file in *$HOME/desktop* (i.e. the user's Interleaf directory) with a *.doc* extension, and removed the original file. The effect is to allow a user to save some Greek text in Interleaf format, then shift to the Interleaf desktop display, where the resulting file appears as a document icon.

We also experimented with providing support for a DOS word processing program which supports a Classical Greek character set, *Nota Bene 2.0*. A table was developed, based on preliminary character assignments supplied by Steven Siebert of Dragonfly Software. A script was added which copied the resulting file to a DOS floppy using the AIX *doswrite* command. The same script could be used to write the results of a search to a DOS minidisk on the RT.

2.7.2. Adding CDFS support

Once we had the HCCP programs running on the RT, which included adding the necessary device support described above, we examined what we needed to do to make these programs work with data on the CD ROM in CDFS format. This was done in stages, as the hardware and device driver problems detailed above allowed us to progress.

We identified the need for an extension to the basic CDFS code which would allow HCCP programs to seek to a specific location in a file. An extended set of system calls were created—*cdfopen, cdfclose, cdfseek, cdfwind, cdftell, cdfread, cdfgetc, cdfgets*—which were designed to act just like the equivalent calls under Unix 4.2 BSD. These were compiled into the CDFS code in July 1986.

At the same time we had to isolate all the file I/O routines in the programs, so that they could be modified to work with either Unix files or CDFS files. This involved restructuring much of the source code. The result was the isolation of the source code for *tlg_sel*, and finally the separation of the source code into two subdirectories: *retr* for modules related to data retrieval, and *displ* for modules related to display.

We assumed from the beginning that the HCCP programs would have to allow for data to be searched either on the CD ROM or on hard disk. This was true simply because we knew there were TLG authors available which were not on the CD ROM. There was also the general goal of creating a single version of the HCCP code which would work in all environments, i.e. with or without data on a CD ROM.

Our first solution was to add support for files on the CD ROM to the existing system. The user's environment contained the value *$GRKLIST*, which represented the name of a structured file listing all the authors available on-line. One of the fields in this file contained the Unix path to the directory containing the text and indices for that author. We had structured the CD ROM so that the root directory (/) contained a directory for each of the authors on the disc. This was another attempt to minimize seek time in CDFS, making all files that needed to be read no more than two levels deep in the file structure. An algorithm was developed whereby the programs look first at the directory field in the *$GRKLIST* file and searched for all necessary files in that directory. If the directory or any file is not found, it then looks on the CD ROM in the root (/) directory of the CDFS filesys-

tem. Using this method we can place corrections to index files for Plato in a Unix files in the directory */usr/TLG/data/plato*, and the programs will look in */plato* on the CD ROM for the text and other files not found on the hard disk.

2.8. Correcting errors in the indices

The question of mixing data on the CD ROM and the hard disk was of more than academic interest. Once graduate students in the Religious Studies department began to make use of the texts on the CD ROM, it was discovered that there were serious mistakes in the indices of Septuagint and Plato. At some point, while generating the word index for Septuagint (the Greek version of the Old Testament texts), a temporary file had overwritten the first three-quarters of the index. The resulting index only contained references to words in the last few works. A similar problem may have corrupted the index to Plato, though in that case, only a small part of the alphabet remained in the index. In both cases, the indices were unusable. A new set of indices were generated from the TLG text file on the RT PC, and the corrected files were placed on the hard disk.

It was then discovered that the indices for several of the later Christian authors—St. Athanasius, St. Basil, Gregory of Nyssenus, and the second parts of Origen and Simplicius—also contained errors. In this case, the words were in the index, but the pointers which located the words in the text were incorrect. After some investigation, it was discovered that sequences of null characters had been used by TLG when coding these texts, a factor that had not been taken into consideration by the indexing programs. These null characters had caused the algorithm used to calculate the byte offset location of each word to lose its count. A simple modification to the initial indexing program fixed this problem. Using this corrected indexing software, a new set of indices for these five authors were generated on the RT PC. Together, the indices necessary to correct mistakes in all seven authors occupied 14 megabytes of space on the hard disk.

The mistakes in these individual author indices were also reflected in the big index of the corpus located on the CD ROM. Since this index takes up nearly 50 megabytes of space, and can only be updated as a single unit, we have had to leave it uncorrected.

At the request of the Religious Studies department we also added the works of Eusebius Caesariensis to the hard disk. This author was made available by TLG after June 1985 in the form of two author files. The indices were generated on the RT PC, and the collection of text and indices accounted for another 10 megabytes of space on the hard disk.

2.9. Isocrates Menu Interface

The HCCP software was originally designed to be used from a standard Unix command line interface. The command for locating a string, *tlg_sel*, takes two arguments: the string to locate and the author in which to search. It returns data to standard output, i.e. the terminal screen. To redirect this data to a file the command must include the Unix redirection character (>) and a filename. The program also allows the user to store a list of strings and/or authors in a file, and pass those filenames in place of a single string or author. To distinguish between a string, an author name, and filenames containing string or author names specific *flags* must be used in the command. This is illustrated in Figure 6 with three examples. In A) the result of a simple search for all the occurrences of *zeus* in the author *homer* is saved in the file *zeus.dat*. To do the same search in B) using a list of words saved in a file called *gods* the *-w* switch must precede the filename *gods*. In C) to extend the same search to a list of authors saved in a file called *attic* the *-a* switch must precede the filename *attic* in the command line. The resulting data file must then be passed to either *tlg_rec*, the program which displays the text on the terminal screen, or *tlg_exp*, the program which expands the references that have been located for viewing or printing. Each of these programs take a variety of arguments as well.

We felt that such a command interface was not suitable for the user population we anticipated for the Isocrates system. Crane had already experienced resistance to this interface among his colleagues at Harvard. He had approached the problem by creating a set of Unix scripts which simplified the syntax needed to perform some of the common tasks one might anticipate. He also built a menu interface, *tlg_new*, which could be used to create word and author files and executing simple searches. This menu system did not address all the possible features of the HCCP software, and so there were still a number of operations which had to be done from the command line.

Figure 6: Example Syntax for *tlg_sel*

```
A)  tlg_sel @zeus homer > zeus.dat

B)  tlg_sel -wgods homer > zeus.dat

C)  tlg_sel -wgods -aattic > zeus.dat
```

We felt that a comprehensive menu interface was required for the system to gain wide acceptance in the departments at Brown. Elli Mylonas took on the task of designing and programming this interface in the fall of 1986. It was developed in C under AIX, and was designed to work on the RT console and other "Greek terminals" supported by the HCCP software. IBM had not offered any support for a "windowing" interface in the AIX operating system, so we were limited to creating a simple menu interface where the user must make his choices by selecting from a list. While we certainly would have preferred to use the RT PC mouse to point to and select items in a menu, building an application with support for this pointing device would have been a complex and time consuming project under AIX. Instead, we allow the user to identify choices by typing a letter on the keyboard.

The first task was to develop a flow diagram of the capabilities of the HCCP software. We identified eight major tasks:

- viewing a list of available authors,
- browsing the work of an author,
- searching for a string,
- creating a file of authors to be searched,
- creating a file of words to be searched,
- viewing a list of existing files,
- viewing an existing data file (a file containing the results of a search),
- and expanding the references in an existing data file.

The top level of the menu interface shown in Figure 7 reflects this list. The search task then branches to the choice of either a simple search or a complex search, the complex option allowing for the location of a word within a specified context of another word. We established several general paradigms to maintain consistency throughout the menus. At any point the user could type *?* and get a help screen which would provide further explanation for the menu they were viewing. They could also type *q* to quit from the program and return to the Unix shell. Typing *r* would return to the previous level of the menu. Commands could be executed one at a time or they could be stacked, so that a user, once familiar with the letters for each option, could move through several menu selections in a single step.

A major function of the menu interface is to maintain a consistent structure to the naming convention used to identify the different kinds of files created by the HCCP software. At different times in a search process, a user may create a data file, which is then used to create a file containing a list of expanded references, which in turn may be saved in any of a number of formats for display on a screen or use with a word processor. The menu interface imposes a set of filename extensions which identify the kind of file be-

Figure 7: Top level menu

```
ISOCRATES - MAIN MENU:

        Type in the letter that indicates your choice.
        You may quit at any time by typing q, and get help
        information at many junctures by typing ?.


        a - List of authors.
        b - Browse.
        s - Search.
        f - Create an author file.
        g - Create a word file.
        l - View list of files.
        v - View an existing data file. (split screen-format)
        e - Expand an existing data file.
        ? - Help.
        q - Quit out of Isocrates.

=======>
```

ing created, and uses these extensions to organize the information created by the user. A data file which is the result of a search for Zeus in Homer becomes *zeus.dat*, while a word file containing variations of the names of Zeus becomes *zeus.wd*, a list of authors to search for references to Zeus would get the name *zeus.au*, and a file containing expanded references from a data file saved in Interleaf format would get the name *zeus.doc* and be placed in the appropriate directory.

The creation of author and word lists also presented a problem. Two text editors were available on the RT PC under AIX: INed and VI. We did not want to require a user to know how to use one of these text editors. On the other hand, we wanted a user who was already familiar with one to be able to use it from the menu interface. The *f* option of the menu for creation of the author or word files (see Figure 8) calls a program which allows the user to type in a word or author one at a time. A second level of the menu allows the user to choose between overwriting the same file with new contents, or appending more words or authors to the same file. The *e* option allows the user to call a text editor to create or modify an existing file of this type. We placed the name of the program to call in a C-shell variable, $EDITOR, located in the user's *.cshrc* file, in order to make the choice of editor a generalized feature.

The more complex problem was how to help the user to build a search request. We solved this problem by creating a natural language search sentence which is interactively modified by selections from the search menu. The search sentence, shown at the top of Figure 9, shows the user information about the search at all times. To build a search the user adds the name of a word or word list, an author or author list, the name of a file in which to save results, and the way in which to use the results of the search. Once this sentence is complete, the user executes the search. To return to example C) given in Figure 6, a user wishing to search for all the words in the word file *gods* in all the authors in the author file *attic* would first use the Create Author File and Word File menus to create these files, then go to the Simple Search menu and supply these files names as the *c* and *e* options. To save the results in a file called *zeus*, this file name is supplied as the *f* option. To make this an indexed search and display the results in the split screen format once the search is complete the default values for the *a* and *g* options are left unchanged. Once these selections have been made the menu would appear as it does in Figure 10.

A useful level of error checking was also added for author names. When the user enters the name of an author as part of the *d* option the program first checks to see that this name is in the current $GRKLIST. If it is not, the menu does not accept the name. Nothing is added to the search sentence and the user is prompted to check their spelling.

The approach taken for the Complex Search and the Expand menus was similar to the one described for the Simple Search. In each case a natural language sentence is displayed at the top of the menu screen, and selections from the menu update the sentence. Once the user has completed the sentence to his satisfaction he executes the command. In this way the menu is able to account for all the possible options available in the HCCP software, as well as check for

Figure 8: Menu for creation of author file

```
ISOCRATES -  MAKE A FILE WITH A LIST OF AUTHOR'S NAMES

         OPTIONS:

         f - give the name of a file that will contain names of authors.
             Filenames may not have more than 8 letters. Do not type in
             the .au extension.
         e - use an editor to create or alter an author file.
         k - permanently erase an author file.
         l - print a list of extant author files.
         v - see the contents of an author file.
         ? - HELP.
         r - return to the Main menu.
         q - quit from Isocrates.
```

Figure 9: Default Simple Search Menu

```
[a]Indexed search for [b|c]?? in [d|e]??. Results will be placed in [f]a
temporary file and will be [g]displayed in split screen format.
-------------------------------------------------------------------------------
ISOCRATES -  PERFORM A SIMPLE SEARCH

 Type in the letter of an option. You may follow it with the letter indicating
 the value you wish to assign to it.

    a -  Linear or Indexed Search. (i, p, s)
    b -  String to search for.
    c -  Name of a file of strings to search for.
    d -  Author to search in or "*" for whole corpus.
    e -  Name of a file of authors to search.
    f -  Name of a data file for the results or "temp".
    g -  What to do with the results of the search.  (e, v, s)
    x -  Execute the specified search.
    ? -  HELP
    r -  return to first Search menu
    q -  quit from Isocrates
```

Figure 10: Updated Simple Search Menu

```
[a]Indexed search for [b|c]the word file gods.wd in [d|e]the author file
attic.au. Results will be placed in [f]the data file zeus.dat and will be
[g]displayed in split screen format.
-------------------------------------------------------------------------------
ISOCRATES -  PERFORM A SIMPLE SEARCH

 Type in the letter of an option. You may follow it with the letter indicating
 the value you wish to assign to it.

    a -  Linear or Indexed Search. (i, p, s)
    b -  String to search for.
    c -  Name of a file of strings to search for.
    d -  Author to search in or "*" for whole corpus.
    e -  Name of a file of authors to search.
    f -  Name of a data file for the results or "temp".
    g -  What to do with the results of the search.  (e, v, s)
    x -  Execute the specified search.
    ? -  HELP
    r -  return to first Search menu
    q -  quit from Isocrates
```

and prevent incompatible options, without the user having to memorize the syntax of each specific command.

The values in these "sentences" are stored as variables by the menu program. Following the suggestions of an early user, we added the feature of having the

values of these variables written out to a file when the user quits from the menu program. In this way, the menu program provides some continuity by "remembering" what the user was searching for from session to session. When users return to the menu program and they move to the search or expand menus, they find the last word or author they were searching for or the last data file they created already selected.

The finished program, called the Isocrates Menu Interface, was documented in a tutorial guide [11] distributed to the users in each department.

3. Delivering the Isocrates System

3.1. Computer usage in the Classics and Religious Studies departments

In June 1985 the Classics department already had several Macintosh and Mac XL (Lisa) computers in the basement computer room used by graduate students, and several Mac XL computers in faculty offices. These computers were equipped with Imagewriter dot-matrix printers. Several of these computers were connected to the BRUNET broadband network via Sytek Localnet 20/100 devices, known as T-boxes. This gave each computer the added ability to work as a terminal on any of the time-sharing computers on campus.

We made available a subset of the TLG corpus on the Sun network at IRIS in the fall of 1985, in order to acquaint the people in the Classics department with the ways in which they could use the TLG texts as soon as possible. This subset consisted of about 100 megabytes of commonly used authors which were then in use on a VAX at Harvard. Crane supplied us with tapes containing directories of text and indices for each author, and complete source code for the HCCP programs. These programs were recompiled on the Sun with no modifications. The HCCP programs already contained support for the Macintosh to act as a "Greek terminal" using the version of MacTerminal 1.0 on the Macintosh developed by George Walsh. Acting as a terminal, the Macintosh could display the TLG texts in Greek.

Interested users at Classics were able to search the TLG texts on the Sun network from any one of the Macintosh computers in their own department connected to a T-box. A special TLG account was set up on the Sun network for use by people in Classics. This account was monitored by Elli Mylonas, who was employed during academic year 1985-86 as the department's computer coordinator. Mylonas was also responsible for training users in how to access the Sun network from the Macintosh, and how to run the HCCP programs. Rudimentary documentation was adapted for the Sun from the documentation provided with the HCCP programs.

At the first public demonstration of the system held at the Computer and Information Services Training Center that fall, we became aware of a group of graduate students from the Religious Studies department who were interested in making use of the TLG texts in their research. Since this department also made Mac XL computers connected to T-boxes available to their graduate students, we made arrangements for them to use the same Macintosh software and the same account on the Sun network as the Classics department.

During this period most of the usage of the TLG data base was made up of requests from faculty members to Mylonas or myself, the two people most familiar with the system, to do searches and return printed results. Before the deployment of the RT PCs to Classics and Religious Studies the Sun workstation at IRIS had the only copy of Interleaf on campus. Since all of the Macintosh computers in these departments used the Imagewriter for printing, only Interleaf could be used to print high-quality Greek on the laser printer.

3.2. Configuration and support of Isocrates in Classics and Religious Studies

In April 1986, discussions were held with Professors Kurt Raaflaub and William Wyatt of Classics to discuss putting machines in that department. At that time it was assumed that the department would be applying to the Faculty Workstation Committee, the administrative body which was in charge of granting RT PC machines to faculty departments. Our inability to deliver the software for supporting the CD ROM players that spring made the timing of the delivery of any systems to Classics uncertain. Later that spring, the Faculty Workstation Project was administratively moved from IRIS to Computer and Information Services (CIS), and the continued general deployment of RT PCs to humanities departments such as Classics was placed under re-evaluation.

A meeting was also held with Professor Stanley Stowers in late August in order to include Religious Studies in the testing of the system. In light of the changes in policy, we came up with an arrangement whereby IRIS would loan an RT PC that was part of the IRIS-IBM Joint Study equipment to each department for the 1986-87 academic year, to support the use and testing of Isocrates. If either department

wished to continue to use the system after that time, that department could request an RT PC from the IBM grant, now being administered by CIS. Both Classics and Religious Studies agreed to this arrangement.

The stipulation was made that both departments had to identify a faculty member and graduate student who would take responsibility for the machines within their departments. These people would be able to attend any of the RT PC training sessions that were being held in September and October as part of the Faculty Workstation Project's deployment to the Music department and the Social Science Data Center.

The two departments were asked to reach an agreement whereby users in both departments would share a single IBM 3812 Pageprinter, to be located in McFarlane House where Classics had its offices.

Classics decided to put their machine in the basement "computer room," a lounge area used by graduate students which already contained the Macintosh computers and terminals used to access the IBM mainframe. This location held the advantage of being generally available to the Classics graduate students, who all have keys to the facility. While it was possible for faculty members to get into this room, it was unlikely that any would travel to the basement to use the system. This lack of access to faculty was a disadvantage, but there seemed to be no good place to put a machine that would encourage *shared* faculty use. Professor Wyatt already had an RT PC in his office, because of his position on the Faculty Workstation Project committee. It was decided that we would put the Isocrates software and a CD ROM player on this machine as well. Two graduate students, Walter Stevenson and Michael Rieder, were selected as the new computer coordinators for the department.

The machine was delivered and installed in October, without the CD ROM hardware. The delivery included the software for searching TLG data on hard disk, and a copy of Homer and Hesiod. An IBM 3812 Pageprinter was put in the basement. The RT PC was connected to BRUNET via a PC Network card, and configured to act as a "print server" for both the machine in Professor Wyatt's office and the machine to be installed in Religious Studies. During the second semester, a connection was made between the RT and a Macintosh Plus on an adjacent desk in the basement room.

Professor Raaflaub had been concerned from the beginning about security in the basement room. Some of the Macintosh software and hardware had been stolen in the spring of 1986, and he wanted to secure the RT equipment in some manner. The RT PC tower was secured to a table and the CD ROM player, which was delivered in December, was locked in a metal box, built for this purpose by the Engineering machine shop.

Religious Studies decided to locate their machine in Professor Stowers' office on the third floor of their building. Timothy Seid, a graduate student who worked with Stowers, was identified as the person responsible for training users in the department. A room was set up on the third floor for graduate students to use the department's Macintosh Plus computers, which had replaced the department's Mac XL that fall. We had discussed putting the RT PC in this room, and putting a Macintosh in Stowers' office, but he elected to have the RT in his office and provide cabling to the two Macintoshes in the graduate student area. This meant that the graduate students could have access to the Isocrates system from the Macintosh, but were locked out of the office containing the RT itself.

The final configuration of equipment for the project consisted of four RT PCs: one at IRIS used for development and testing, one in the basement of Classics, one in Professor Wyatt's office on the second floor of McFarlane House, and one in Professor Stowers' office in Religious Studies. The hardware and software of the machine in the basement of Classics is illustrated in Figure 11. The machines in Professors Wyatt and Stowers' offices were able to share the 3812 Pageprinter by using the Brown University Network Services (BUNS) software, developed by the Faculty Workstation Project staff. This software allowed WPS to use a printer on a remote machine. The same software also provided 3270-terminal emulation at channel speeds for access to CMS on the IBM mainframe from each RT PC, and allowed us to do remote listings from and copies to any machine across the network.

3.3. Performance Issues

From the beginning of the project we were apprehensive about how the HCCP software would perform once the data was on CD ROM. We knew that reading data from a CD ROM would be slower than reading data from hard disk, and faster than reading data from floppy media. We knew that head positioning on CD ROM players was slow, and so minimizing the number of seeks required to read a file would speed up the application.

Performance is a completely relative issue. What is fast and what is slow depends entirely on what we are comparing it to. If we say we want to locate all the words beginning with the Greek stem meaning "maker" in the complete works of Aristotle, and print

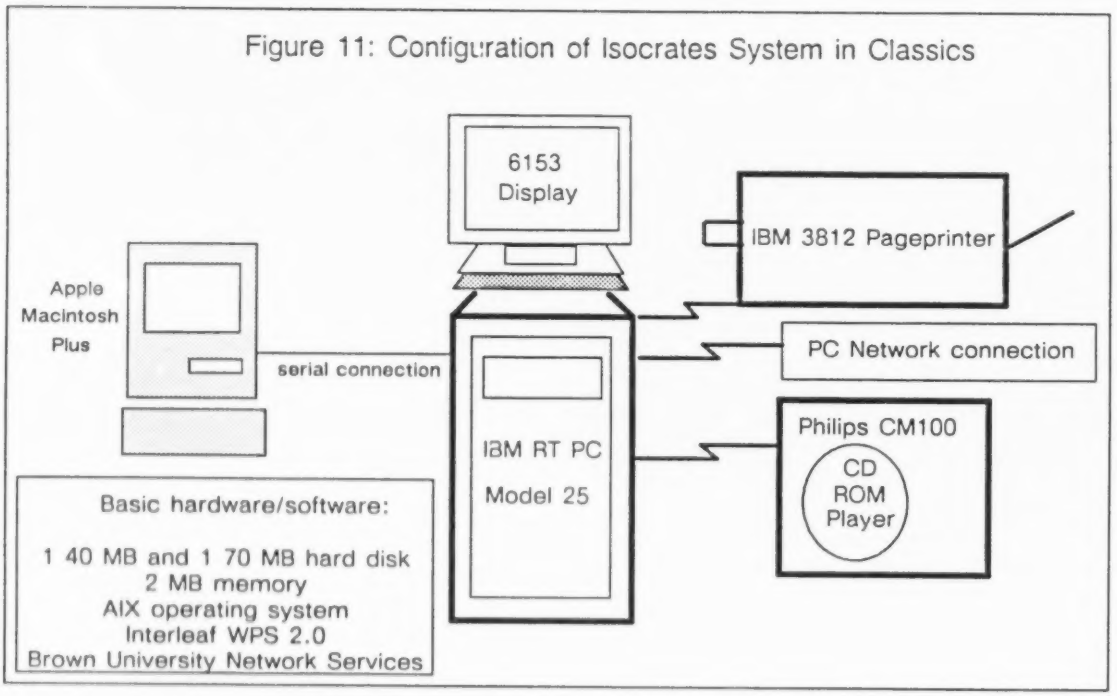


Figure 11: Configuration of Isocrates System in Classics

a list of these words and their locations along with five lines of context for each occurrence, what is "acceptable" performance? Isocrates finds the 32 forms of the stem in about 30 seconds. It takes another 20 minutes to put the 479 references with 5 lines of context into a file, and another 15 minutes to print the results. To do this using printed concordances, research assistants, and retyping all the results with a word processors could take weeks or months.

We have not sought to collect performance data on other computer systems which can be used to access the TLG texts. There are implementations of the HCCP software on other Unix platforms with some portion of the TLG texts on hard disk. The Duke University Humanities Computing Center has put together a system which allows the search and display of some TLG texts on the IBM PC/AT. Ibycus Systems sells the Ibycus minicomputer and the Ibycus Scholarly Computer with CD ROM player, used at the University of North Carolina, University of Pennsylvania, Princeton, and Yale, which also provides a method for searching and displaying the TLG texts [6]. We do not have, nor have we sought, performance data on any of these systems with which to compare our experiences.

The goals of the Isocrates Project were to demonstrate the use of CD ROM on the new workstation, create a productivity tool for scholars in this field, and test a CD file system in a real application. The entire system has performed well enough to be accepted by some members of both departments. We lack any comprehensive and objective data on how the system has been used thus far, though we do have informal user response. The areas we have evaluated are those where we can now see that what the system can do differs from what we hoped it would do. These are the areas discussed below.

3.3.1. Evaluating CD ROM performance

We have prepared figures which compare the response time for the basic HCCP applications on the RT PC when searching an author on hard disk versus the same search off CD ROM. These times were collected on the RT PC running version 1.1 of the AIX operating system. The times for the CD ROM reflect the efficiency of the current CDFS implementation on the RT, as well as the VRM/AIX device driver for the Philips CM100/CM155 hardware. The programs were run with the *time* command built into the C-shell (csh). The numbers shown for **hard disk** and **CD ROM** are the elapsed or "real" time, as reported by this command. The number of **hits**, i.e. unique words found by the search, is shown in the last column. In each case we ran the same command several times and recorded the average response time, since the "real" time does vary depending on the state of the machine, the location of the device driver in memory,

and what files have been cached by the previous command. These times are rounded to the nearest second. The numbers are shown in Figure 12 below.

The first example is an indexed search of Septuagint; the text occupies about 5 megabytes, and the search locates two words. The second example is an indexed search of the Homeric Hymns; this text take up just over 100 kilobytes, and the search locates six words beginning with the stem δημητ. The third example is a similar search of Homer; the text is about 1.5 megabytes and the search locates one form of Ζευσ. A similar search for forms of the goddess Demeter locates three forms. These examples illustrate how the efficiency of the indexed search in the HCCP software is independent of the size of the text. The search may take longer when more words are found, but is not significantly effected by the size of the index itself.

When the stems for Zeus and Demeter are placed in a file called *gods* and the same search is repeated in Homer, the search times reveal one of the performance problems on the CD ROM. The 12–16 second difference in the first four examples between the hard disk and CD ROM times are attributable to two factors. The first is that CDFS has about a 10 second overhead to locate and open the author directory. The second factor is the relative slowness of a binary search on the CD ROM which requires any reposition-

Figure 12: Comparison of hard disk and CD ROM access time

command	hard disk	CD ROM	hits
1) tlg_sel @logos septuagint	0:02	0:14	2
2) tlg_sel @dhmht homer_hymn	0:02	0:18	6
3) tlg_sel @zeus homer	0:02	0:18	1
4) tlg_sel @dhmht homer > demeter1	0:02	0:18	3
5) tlg_sel -wgods homer	0:04	0:26	4
6) tlg_sel -aauths @dhmht > demeter2	0:04	0:30	9
7) tlg_rec demeter1	0:04	0:15	1
8) tlg_exp demeter1	0:02	0:12	6
9) tlg_exp demeter2	0:03	0:17	17
10) tlg_exp -v51 -dileaf demeter1	0:10	0:25	6
11) tlg_exp -v51 -dileaf demeter2	0:25	0:51	17
12) tlg_sel -p @logos	n/a	0:26	3/103
13) tlg_sel -s @kai@nu homer_hymn	0:05	0:25	5
14) tlg_sel -s @kai@nu septuagint	1:44	9:27	423

ing of the read head. We see in example five that the hard disk time for finding these two words in a single author is equal to the sum of the times for example three and four, while the increase in the CD ROM time is significantly less than the sum of the previous times. This difference is illustrated further in example six, where two authors, Homer and Homeric Hymns, are placed in an author file, *auths*, and the search for forms of Demeter is run again. This requires the opening of two author directories. The hard disk time still represents the sum of examples two and four. The CD ROM time remains less than the sum of examples two and four, but slower than the time for finding two words in a single author. More time is used on the CD ROM locating a second author than locating a second word in the same author.

The next several examples illustrate this point further. In example seven the *tlg_rec* program, which reads the data file created by *tlg_sel* in example four in order to locate and display the first form of Demeter in the works of Homer, takes approximately 11 seconds longer to do this job off the CD ROM than off the hard disk. The *tlg_exp* program in examples eight and nine shows the times for expanding the references located by *tlg_sel* in examples four and six. This program follows a pointer in the data file and expands it to the location of each occurrence of the word expressed as an absolute offset in the file. When *tlg_exp* is run on the same data with options to extract five lines of context for each occurrence (*-v5l*) and translate the results into Interleaf format (*-dileaf*) in examples ten and eleven, the "real" time increases with the complexity of the data at the same rate in either hard disk or CD ROM. These examples add more disk I/O, which is slower on the CD ROM, but take most of their time translating the data that has been extracted, which is independent of disk access.

Example twelve illustrates using *tlg_sel* to search for λογοσ not in a single author or list of authors but in the big index of all words in all authors. We did not have room to put this index on hard disk for a comparison, but the results on the CD ROM illustrate how this kind of search is relatively "fast" in any case. This search creates a file containing references to 3 forms of λογοσ in 103 authors in 26 seconds.

Examples thirteen and fourteen illustrate linear rather than indexed searches. These involve reading the entire TLG author file itself in a linear fashion. In example thirteen, we have searched for the prepositional phrase και νυ in Homeric Hymns, which located 5 occurrences. A similar search in Septuagint located 423 occurrences. The linear search of Septuagint was about twice as "slow" as that of Homeric Hymns by comparison with the same searches off hard disk. Since I/O is much slower on the CD ROM and the linear search is very I/O intensive, the "slowness" of this kind of search increases with the size of the text being searched.

To look for a "worst case" example, we ran several linear searches of all 178 authors on the CD ROM, looking for a common preposition. This requires the software to do a linear read through every text, and report the results to a file. We found that it takes *tlg_sel* about 6 hours to complete a linear search through all the texts on the CD ROM. We did not have a similar figure for doing a linear search on a hard disk system.

3.3.2. Sharing the CD ROM device

What is more problematic than the 10-12 second overhead for locating an author directory on CD ROM is the way in which the AIX/VRM device driver must lock out all other processes while a read is taking place.

Our original intention was for Isocrates to be used as a multi-tasking/multi-user system. While there is no "window" system under AIX 1.1, the operating system does provide an interface for what is called *virtual terminals*. AIX allows a user to have several virtual terminals open on the console (i.e. the RT PC's primary monitor) at the same time, each taking up the entire display area of the monitor. The user can then move, or *hot key*, from virtual terminal to virtual terminal, in rotation, by hitting a key combination. This resembles the Macintosh Switcher program, with one important difference: since each virtual terminal represents a Unix shell or application program, the operating system can handle each program running simultaneous tasks. This means that the menu interface to the HCCP programs can be running in one virtual terminal while another is used to run the Interleaf WPS program. We anticipated that a single user on the RT console might wish to go back and forth between a search and their writing, leaving a long search running in background while continuing to write.

The AIX/VRM device driver we wrote does not allow this to happen. Because the hardware required programmed I/O on the CM155 interface card, the device driver had to block all machine activity during a read to ensure successful timing. Other AIX applications do get momentary attention from the processor. The effect of a read from the CD ROM on the applications running on other virtual terminals on the console, or other terminals connected to the RT via serial lines, varies from the disabling of keyboard input to the sluggish response of an overloaded machine.

In Classics, where the graduate students had access to the RT and Macintosh in their basement computer area, there has not been enough demand for using the RT to make simultaneous use of the data base on the RT console and the Macintosh a significant issue. This connection between the two machines was used primarily to transfer Greek text files saved in GreekKeys format on the RT to the Macintosh for word processing. In Religious Studies, the graduate students were limited to accessing the data base from a Macintosh, while the RT was used by a faculty member in his office. Even in this case (one faculty member, one graduate student) usage was not heavy enough to make simultaneous use of the data base an issue. It remains to be seen what effect heavy usage of the system by more than one user might have.

3.3.3. Interleaf WPS

We had hoped that Isocrates would be capable of integrating the TLG text with a document preparation system appropriate for use by scholars who are preparing journal or book manuscripts. Interleaf seemed like a good candidate for the kind of document preparation system needed for this job. Being a product designed for the technical publishing market, it clearly had great strengths. Its user interface and structured document design were very important. Its way of displaying the text of each page on the screen with actual fonts and page layout was also important, since this meant the user could see the English and Greek text as they composed their document. Its ability to mix text and graphics in a document was not appreciated by our users, since graphics has no place at present in Classical scholarship. There were weaknesses in WPS version 2.0, the version of Interleaf available for the RT PC, which limited its use in the two departments. We were aware of these weaknesses before the project began, as they were pointed out in a Faculty Workstation Project memo [8] in the fall of 1985 which evaluated the Interleaf software in the context of appropriateness for general deployment on campus.

WPS 2.0 does not provide support for a full European character set. There is no way to put simple accents over letters in any font, other than the Classical Greek font we created. This means that the inclusion of a German or French word which requires an accented vowel is beyond the capability of the software. Lacking any dead key facility, it also means that there is no reasonable way to access character/accent combinations from the keyboard. While users can type accented Classical Greek, they must use a large keyboard map to locate the key position for each character/accent combination. Users in Classics had already adapted to typing accented Classical Greek using either the dead key system on the Macintosh or a coding system used for Script on the IBM mainframe. Both of these systems involve learning a key or character to represent an accent in combination with the key or character to represent the vowel. Users found the process required to do the same thing in Interleaf—shifting to the italic font and hunting through a large keyboard map for just the right position on just the right key for just the right vowel/accent combination—far more complex.

WPS 2.0 does not provide a reasonable way to do on-page footnotes. The method that is available—putting footnote text in a graphic frame which must be manually sized for each page—can be used only in very simple cases. There is no auto-numbering feature for footnotes or any other kind of number list. This makes the software inappropriate for producing scholarly books or articles. It also makes it inappropriate for producing scholarly graduate papers. More than any other weakness, the lack of a reasonable footnote feature has prevented any of the graduate students from making use of WPS for preparing their own papers.

WPS 2.0 on the RT PC does not provide a way to print on any device other than the IBM 3812 Pageprinter. While this 240-dpi output is good for some purposes, actual typeset output is required for most major projects. Interleaf does support typesetters from other versions of their software on other hardware platforms, but there is no support for higher-quality output devices from the RT PC, and no service bureau currently accepts Interleaf source files for typesetting on any other device. Even if they did, additional work would need to be done to create a matching Classical Greek font for that output device.

In 1987 Interleaf issued two new releases of their product. The first, version 2.5, added a number of features but did not correct any of the weaknesses described above. This was released for the RT under the name RT.PS. They then issued version 3.0 of their software, which became available to educational institutions as the University Publishing System (UPS) product, though only on the Sun and Apollo hardware platforms. This version corrects many of the weaknesses found by WPS users at Brown, with the important exception of support for accented characters. Due to marketing considerations, Interleaf has not chosen to make this version available for the RT PC. Yet another version of the Interleaf product, which addresses support for modern European languages, has been demonstrated at trade shows but is still under development.

Since the project began a number of document preparation systems have been developed for the Macintosh and MS/PC-DOS computers which can provide much of the functionality needed by scholars in Classics. Both *Microsoft Word* 3.0 on the Macintosh and *Nota Bene* 2.0 on the MS/PC-DOS machines, programs which support a full Classical Greek character set, have good keyboard support for accented characters, support complex on-page footnotes, and output to a variety of high-quality printing devices. Due to marketing considerations we have no reason to believe that either of these products will become available under AIX. Though neither of these programs is currently being used by people in the Classics or Religious Studies departments at Brown, Isocrates can be used to save text for word processing in either of these systems.

3.4. Current Usage

Usage of the TLG data base has not been formally monitored during the project. What we know of the volume and type of usage is derived from informal interviews and other contacts with users in the departments.

During the major period of software development at IRIS (September 1985-September 1986), access to the TLG data base was limited to Macintosh "Greek terminals" accessing the Sun network and no comprehensive menu interface. We did whatever we could to encourage faculty and graduate students to use the system. The primary users of the system were graduate students with specific projects or assignments. For example, graduate students in a course on Aristotle, taught by Professor Martha Nussbaum, were encouraged to do searches of the Aristotle corpus. A number of searches were done for members of the Classics faculty by Mylonas, which resulted in printed reports locating usage of certain terms in an author or group of authors. A graduate student in the Department of Linguistics made extensive use of the system to generate examples for her doctoral thesis, which compared the usage of specific Classical Greek verb forms in two time periods.

Beginning in October 1986, when the RT PC equipment was delivered to the departments, the responsibility for encouraging usage was transferred entirely to the computer coordinators in Classics and Religious Studies. Data was at first limited to a few major authors on the hard disk, while we continued to debug the CD ROM software and finish the menu interface. The system was already very much in use when the CD ROM was added in December. The performance of the software remained unstable, however, through February 1987.

Usage by graduate students in both departments continued to grow during the second semester, as the device driver was stabilized and the menu interface completed and refined. We estimate that approximately ten graduate students in Classics and another three in Religious Studies used the system on a regular basis during the last semester of the project. Students in Classics seminars on the literature of the 4th century A.D., Roman history, and Homer all did searches of texts on the CD ROM which related to their courses. Most of this was done on the student's initiative, though Professor Wyatt gave the members of his Homer seminar the general assignment to make use of the system. In Religious Studies a seminar on *4 Maccabbees* and another in early Christian writers led to a great deal of searching in those areas.

3.4.1. Types of use

In general we have found that the Isocrates system is used for three purposes: as a concordance, as a lexicon, and as a source of texts.

The most immediate use of the system is as an electronic concordance. Locating a word using an indexed search is easier than using a printed concordance. In addition, no printed concordance exists for most of the texts on the CD ROM. The applications found for this concordance feature are various. Students have been able to confirm or refute the proposed date for certain texts by locating specific terms in a text. A graduate student was able to locate a comprehensive list of text which appear to be funeral orations by searching for specific terms associated with that genre. Professor Hossein Ziai of Religious Studies supplied a graduate student with a list of Greek philosophical terms which he felt had some influence on developments in Islamic mysticism. The graduate student was able to generate a comprehensive list of references from these words, identifying who used each term and how they were used. The linear search also allows the user to locate phrases, and this feature, while it is more time consuming, has proven particularly valuable. Professor Stowers was able to gather usage information throughout the corpus on a significant phrase which occurs in the writings of St. Paul [14].

The system was also used to augment definitions found in the standard Greek-English lexicon. The definitions in such a lexicon are based primarily on examples of usage. The major difference between the small, medium, and large lexica used in this field are the number of usage examples quoted with each term. Graduate students in Classics found that they could

extend their understanding of a word by searching for the word on the system and reading through the examples.

The system has also proven to be a useful source of texts. It has been used to locate and then copy, from the CD ROM to a word processing format, portions of a text being studied, for the purpose of further annotation. In particular, graduate students copied passages from *4 Maccabbees* from the CD ROM into MacWrite format on the Macintosh, in order to reorganize, translate, and annotate the texts. In at least one case, the electronic version of a minor author, Hermogenes, proved to be the only fair copy available on campus. In the case of the Religious Studies department, where the original text of most of the Classical Greek authors are not kept in the departmental library, it was more convenient for graduate students to browse and read a text off the CD ROM than to go to the campus library and locate a printed edition.

Our original hope—that Isocrates would be used for the preparation of major papers and manuscripts—has not been fulfilled in practice. In a larger context—the Faculty Workstation Project's deployment of RT PCs to several departments on campus—Interleaf WPS has won acceptance from most, if not all, faculty users. The type of usage has not been as a scholarly document preparation system, however. In the case of Classics and Religious Studies, this usage is limited to short documents such as correspondence and class handouts. WPS is used by the two faculty members with primary access to the RT PCs and by graduate students in Classics, who have access to the machine in the basement of that department. Users in both departments cite two major weakness of WPS as the reason for their limited usage: lack of on-page footnotes and the difficulty of typing Classical Greek, both of which are described above in section 3.3.3. By way of an alternative, all the Classics graduate students continue to use Waterloo Script/GML under CMS on the IBM mainframe for the preparation of major papers. While the Greek that they type using this system appears in the form of coded roman characters on the terminal screen, the version of the Script software at Brown supports printing in a Classical Greek font on the Xerox 9700 laser printer. The Classics faculty also uses this method when employing a computer to prepare large manuscripts, usually with graduate student assistance. We do not expect this pattern of usage to change in the near future. Despite their other investments in Apple Macintosh equipment, neither the Classics nor the Religious Studies departments currently plan to purchase a laser printer such as the Apple Laserwriter, which would support one of the Postscript Classical Greek fonts now available.

4. Conclusions

4.1. Realizing the original goals

In reviewing the original goals of the project, we can draw the following conclusions.

We have been able to demonstrate the use of an interactive application (the HCCP software) for accessing a large data base (the TLG texts) on a CD ROM within the workstation computing environment (the IBM RT PC running AIX). This demonstration has proven to be a useful system for members of two academic departments, and has provided a useful "thought piece" for members of other humanities departments not directly interested in the particular data base chosen for the project. The TLG data base has proven to be a good choice for several reasons. In terms of size it is large enough to be too big for delivery using local magnetic media, but small enough to fit within the current CD ROM standard of 550 megabytes per disc. There is good application software available for accessing the data base, and the addition of a comprehensive menu interface has made the application generally useful to interested scholars in this field. The RT PC has proven to be an adequate platform for delivering this application. We encountered unexpected complications in dealing with the RT PC bus hardware interface, and in developing device driver software for the VRM/AIX operating system.

If we have learned anything generic from this development process, it is that much can be gained by adapting and adding to existing software projects. By working with elements developed separately at a number of campuses we were able to create a very useful system with a minimum of personnel resources. To have tried to duplicated the enormous efforts already expended by our colleagues at Harvard, M.I.T., and U.C. Irvine would have been foolish, as well as impossible. On the other hand, these projects in their current form were not useful to the users at Brown. Our efforts to integrate, synthesize, and improve these *pieces* into a working system have been successful.

We have also learned to avoid getting involved in the development of device driver software whenever possible. The difficult task of interfacing a new hardware device to an unfamiliar operating system is unpredictable, regardless of the local expertise available. This task is complicated further when the hardware device and the operating system are not made by the same company. In future development projects which involve CD ROM, we will make the effort to assure greater cooperation on the part of the hardware and software manufacturers involved. In particular, we will

not work with a CD ROM drive on a specific hardware platform without device driver software supplied by one of the manufacturers.

Our success in the area of creating a productivity tool for the scholar working in this field has been only partial. As is discussed below, far more work needs to be done before we can have an adequate empirical model of what it is that scholars' do, and how we can best design tools to help them in their work. We have been able to demonstrate how a close integration of a document preparation system with a full-text data base could be useful to scholars in this field. The best document preparation system available to us on the RT PC has not proven to be the "best tool" for the Classics scholar. Productivity could be increased by providing additional access points to the information already in a data base such as the TLG, and by further work in the user interface design. Both of these subjects are discussed below.

The Isocrates project has given us an opportunity to test the Compact Disc File System (CDFS) with a full-scale application. The CDFS software has proven to be a good medium for pre-mastering data for a CD ROM. It's performance, i.e. its ability to locate and deliver data from CD ROM in a timely fashion, has proven to be adequate for this application. Placing the data base on CD ROM has proven to be a successful way of delivering this kind of full-text application to users in their own department. The CD ROM and CD ROM player are quiet, inexpensive, and unobtrusive.

4.2. Areas for further work

4.2.1. Suggestions for extensions

We expect that more work will be done at other universities in the coming year to add to and further adapt the TLG data base and the HCCP software to other work environments. An increasing number of people interested in these texts have already invested in single-tasking personal computers such as the Apple Macintosh and the IBM PC/AT. Now that CD ROM device drivers are available for both these computers, we expect to see efforts to port the HCCP software to the Macintosh and PC/MS-DOS operating systems in the coming year. Because of the hardware and operating system independence of the CDFS design, the **TLG Pilot CD ROM #B for Experimental Purposes** could prove useful in either of these cases. However, no further work in this area is planned at IRIS.

We have developed several suggestions for extensions to the capabilities currently offered by the HCCP software and the TLG data base.

An effort should be made to integrate the information currently in the TLG Canon with the texts themselves. The Canon should be distributed as a data base, and linked to the texts so that when looking at a text, the user will have access to the bibliographic information about that text. Being made aware that the line I am looking at is "il 5 217" is not enough. I should be able to see the information about it in the Canon: that this is the work of Homerus Epic., 8 B.C., and that this passage is from *Ilias*, ed. T.W. Allen, *Homeri Ilias*, vols. 2–3, Oxford: Clarendon Press, 1931: 2:1–356; 3:1–370. Without offering any comments on how the Canon data base is currently structured, we suggest that the user should also be able to build a query by using the information associated with individual authors. So, for example, I should be able to search for all the occurrences of a term in only those authors with the generic epithet *Phil.* who wrote between the 4th and the 1st century B.C. It would also be useful to have information on those authors in the Canon whose works are currently not available in the TLG data base. To extend the previous example, a search of all philosophers between these dates should also return a list of those authors in the Canon who meet this criteria but whose work cannot be searched in this manner.

Additional work is also called for in providing access to the sorted word lists that are already a part of the HCCP indices for each author. These lists are a resource barely utilized by the current software. The user should be able to browse these word lists in a manner analogous to browsing the works of an author. The user should also be able to build a search by selecting a word or word stem from a list, rather than having to guess whether a particular word, spelled out in roman transliteration, will appear in that author.

4.2.2. Further work on CDFS

CDFS continues to evolve. A smaller subset of the original design which was limited to read-only media, named Micro Compact Disc Files System (MCDFS), was developed by Garfinkel at the Media Lab during 1986 and implemented on the IBM PC/AT. Work is going on at M.I.T. to adapt CDFS for use on the write-once optical drives that are becoming available. At IRIS we hope to add support for the Sun Network File System (NFS) to CDFS, so that application programs will no longer have to know the difference between a CDFS file and a file in the Unix file system.

This difference will be handled by NFS on the operating system level.

4.2.3. The scholar's working environment

When you give scholars in a field such as Classics a large textual data base like the TLG, along with application software for accessing it in some manner, you quickly learn that this is only one small piece of a much larger information need. As massive as the TLG data base is (and it does represent one of the largest collections of text in any field in the Humanities at this time), it is *only* Classical Greek literature. The Classicist is also concerned with all of Latin literature, as well as literature in other languages such as Hebrew, Coptic, Aramaic, Syriac, Persian, Arabic, etc., which may be related by subject or historic proximity. In a field such as Classics, there is also specialization in areas such as Classical Art and Archaeology, Epigraphy, and Papyrology. Each of these suggest their own large data base needs. These needs are not unique and unrelated, but they are all different in important ways.

It seems clear that, in general, the scholar needs access to four classes of data resources. These are:

- *Primary text/data/image collection(s) in the field*: collections of material which are themselves the subject of study.

- *Language/image reference material*: collections of information that are used to analyze or better understand primary material.

- *Bibliographic citation and reference material*: collections of information describing original source materials and secondary material.

- *Personal information resources*: collections of annotated bibliographic information and other data selected for personal use, as well as other forms of cross referencing and annotation materials.

The TLG data base represents an important part, but only a part, of the primary text collection in the field of Classics.

The text collection for each Classical language will also require its own on-line lexicon. While a great deal of research has been done in recent years concerning on-line spelling aids, dictionaries and grammar aids in English, we are not aware of major work being done on electronic lexica in Classical languages. In addition to electronic versions of Greek-English and Latin-English dictionaries, other reference language and image reference material will be needed. Data bases such as the Lexicon Iconographicum Mythologiae Classicae (LIMC), an index of Classical Iconography currently under development at the State University of New Jersey at Rutgers [13], must be integrated into the Classical scholar's working environment. Special reference encyclopedias, such as collections of information on Classical person and place names or Greek and Roman mythology, should be available in an electronic form.

The scholar will also need access to a comprehensive bibliographic data base of secondary work in his or her field. Such bibliographic data bases have already become an invaluable part of medicine, law, physical science and engineering due to the importance of timely access to information in these fields. In a Humanities field such as Classics, where the oldest, rather than the most recent, article on a particular subject may be the most important, most significant bibliographic data bases have yet to be created. Work is currently underway "to explore the mechanics of computerizing *L'Anne Philologique*" [13], the core journal in Classical Philology. Once it becomes available, this too will become a required part of the system.

Access to all of this information in one form or another underscores the need for the fourth class of data resource listed above: the personal information management system, or personal data base. The form that such a system should take is an important area for research. A recent study of the work environment of Art Historians illustrates just how little we actually know about how scholars work [1]. Tools must be devised which will help the scholar manage an electronic information network that mirrors and augments the special knowledge in his or her field.

All this suggests that such a system must be capable of presenting a massive amount of information in a simple and well organized fashion. Further research is needed in the area of user interface design to explore the problem of presenting a uniform interface to the variety of applications to search and manipulate different types of data base resources. Replacing command-line syntax with menu selections that build a search sentence is a small step down a long road. That road must lead us to a new model for the man-machine interface capable of representing the complex information needs of the Humanities scholar.

Acknowledgments

A significant number of people have worked on the *Isocrates* project since its inception in June 1985.

The production of the **TLG Pilot CD ROM #B for Experimental Purposes** is the work of the *Thesaurus Linguae Graecae* (TLG) at the University of California, Irvine, under the direction of Theodore Brunner, and was supported by a grant from the David and Lucile Packard Foundation. The data base itself, that is the machine-readable version of Greek authors, is the result of over a decade of data entry, verification and maintenance by the staff of the TLG, and is the property of the TLG. It is used by Brown University under license from the TLG. Any distribution or use of the CD ROM used for this project is at the discretion of the TLG. IRIS's role in the production of the **TLG Pilot CD ROM #B for Experimental Purposes** was limited to the creation of the index files and the formatting of the data for pressing.

Encouragement and partial funding for the project was provided by IBM Academic Information Systems (ACIS), with the help of Les Comeau.

Interleaf, Inc. provided a specially licensed copy of the Interleaf Font Editor to us, along with bitmaps of their Greek fonts, so that we could create the necessary Classical Greek fonts. Thanks are due to Mike Cottle and George Potter for making this possible, and to Dave Anderson, Jim Crawford, and Kathy Nitchie for providing software, advise and support.

Much of the programming work for *Isocrates* was done by Gregory Crane, Assistant Professor of the Classics at Harvard University. Additional programming and technical assistance was provided by many people at IRIS: Scott Anderson, Mike Braca, David Bundy, Brian Chapin, Arif Diwan, Sam Fulcomer, Bern Haan, Michael Pear, Donald Wolfe.

William Wyatt and Kurt Raaflaub of the Brown Classics department have provided advice and support to the project. Elli Mylonas, computing coordinator for that department, has provided invaluable assistance in training and supporting end-users of the system, as well as designing and programming the menu interface. Additional support in the Religious Studies department was provided by Timothy Seid and Professor Stanley Stowers.

Support for working with the Compact Disk File System came from Simson Garfinkel and Professor Walter Bender of M.I.T.'s Advanced Media Lab. Additional support for working with the CD ROM and CD ROM player was provided by Rus Gant of M.I.T.'s Project Athena, and Michael Johnson of Project Athena/IBM. The work of a number of people from Philips Subsystems and Peripherals has made it possible to interface the CM100 CD ROM player to the IBM RT PC, among them Larry Schwartz, Bert Gall, Keith Moore, Robert Moes and Robert Van Eijk.

References

1. Beeman, William O., Elizabeth Bakewell, Carol McMichael Reese and Marilyn Schmitt. *Object, Image and Inquiry: The Art Historian at Work.* (Los Angeles: Getty Museum, 1987)
2. Berkowitz, Luci and Karl Squitier, editors. *Thesaurus Linguae Graecae Canon of Greek Authors and Works.* (Oxford: Oxford University Press, 1986)
3. Garfinkel, Simson. *A File System for Write-Once Media.* unpublished. (Cambridge: Advanced Media Lab, Massachusetts Institute of Technology, 1986)
4. Harvard Classics Computer Project. *Indexing TLG Author Files.* (Cambridge: Department of the Classics, Harvard University, 1987)
5. Harvard Classics Computer Project. *User's Guide for Search Programs.* (Cambridge: Department of the Classics, Harvard University, 1987)
6. Hughes, John J. *The IBYCUS SC: A Multilingual Computer System for Scholars.* (Whitefish MT: *Bits & Bytes Review* Vol. 1 No. 1, October 1986)
7. Kahn, Paul. *CD ROM Proposal to IBM.* unpublished. (Providence: Institute for Research in Information and Scholarship, Brown University, 1985)
8. Kahn, Paul, and Lynne O'Brien. *Evaluation of Applications for the Advanced Function Workstation.* unpublished. (Providence: Institute for Research in Information and Scholarship, Brown University, 1985)
9. Kahn, Paul. *Isocrates: Greek Literature on CD ROM*, in *CD ROM: The New Papyrus.* (Seattle: Microsoft Press, 1986)
10. Kahn, Paul. *A Description of Isocrates: All of Greek Literature on a CD ROM*, in *Proceedings of the Conference on Computing and the Classics, July 1986.* (Los Angeles: UCLA Department of Classics, 1987)
11. Kahn, Paul. *Isocrates Menu Interface: A Quick Tutorial.* (Providence: Institute for Research in Information and Scholarship, Brown University, 1987)
12. Johnson, Michael. correspondence with the author.
13. Small, Jocelyn Penny. *Databases for Research in Classics.* Talk at the Annual Meeting of the American Philological Association, December

1986. (New Brunswick NJ: Lexicon Iconographicum Mythologiae Classicae U.S. Center, 1986)

14. Stowers, Stanley. *Review of "2 Corinthians Chapter 8 and 9: The Hermeneia Commentary" by Hans Dieter Betz.* (Journal of Biblical Literature, 1987 forthcoming). Also see *"Peri men gar" and the Integrity of 2 Corinthians Chapter 9.* (article submitted to Journal of Biblical Literature).

15. Yankelovich, Nicole. *INTERMEDIA: A System for Linking Multimedia Documents.* IRIS Technical Report 86-2. (Providence: Institute for Research in Information and Scholarship, Brown University, 1986)